4D v11 SQL の XML によるストラクチャの読み込み/書き出し

Atanas Atanassov, Technical Services Team Member, 4D Inc.

Technical Note 08-26

概要

このテクニカルノートでは、4D v11 SQL の新しいストラクチャ書き出し/読み込み機能を紹介します。4D v11 SQL Release 1 は XML ベースのストラクチャ操作サポートを提供しました。この新しい機能を使用すればデータベースストラクチャを書きだし、そこから新しいデータベースを作成したり、あるいは XML ストラクチャデータをデータベース間でコピーすることすら可能です。

XML についてまだよくご存じない方のために、このテクニカルノートでは 4D における XML の基本についても説明しています。

はじめに

データベースストラクチャ (ストラクチャ定義) を XML ファイルに書き出せます。そしてこの XML ファイルを使用して新規の 4D データベースを作成できます。この機能は 4D v11 SQL Release 1 以降に実装されています。

Note: *ストラクチャ定義をバックアップやデータ復旧ルーチンで使用すべきではありません。ストラクチャ定義はテーブル、フィールド、インデックス、リレーション、ストラクチャエディタ設定、そしてストラクチャの完全な定義に必要な属性から構成されます。ストラクチャ定義はデータベースのデータファイルには影響を与えません。XML ストラクチャファイル内には一切データが含まれていないことを覚えておいてください。*

すべての書き出されたオブジェクトは XML ストラクチャ内で独自のタグを作成します。それらがすべて集まって、ストラクチャ定義となります。

例:

```
<root>
   <table>
          <field>
          . . . . .
          </field>
   </table>
   <index>
   . . . . .
   </index>
   <relation>
   . . . . .
   </relation>
</root>
```

新規データベース作成に加え、ストラクチャ定義ファイルを XML をサポートする HTML など他の技術と共に使用できます。この場合 XML スタイルシートの一部である XSLT (XSL Transformation) ランゲージを使用します。

このテクニカルノートでは、4D ストラクチャ定義を完全理解するために重要なシンタックスルールを含む XML ランゲージの基本をカバーします。4D の XML コマンドを概観し、4D が XML ファイルを作成し処理する方法を見ます。そして 4D がストラクチャ定義を作成する方法と、どのように使用するかを説明します。

XML について

Extensible Markup Language (XML) は急速に発展した技術で、データの管理や表示、組織化に使用します。これはデータ交換において特に使用され、データとデータ構造定義にタグを使用します。XML ファイルはテキストエディタで開いて操作できます。また XML は厳密なシンタックスルールに基づきます。

- ルート要素を持ちます。
- カスタム定義の開かれたタグは必ず閉じるタグとペアでなければなりません。
- XML は大文字小文字を区別します。開くタグと閉じるタグは大文字小文字が一致していなければなりません。
- タグはお互いに正しくネストしていなければなりません。
- すべてのタグ属性値は引用符で囲まれていなければなりません。
- XML 要素内で"<"や">"を使用してはいけません。パーサはこれらの文字を要素の開始や終了として認識するため、エラーが生成されます。これらの文字を使用する場合、実体参照 "<"や">"を使用します。
- コメントが使用できます: <!—コメント -->

例:

```
<?xml version="1.0" encoding="utf-8"?>
<name>                  // ルート要素
<first>John</first>     // 開くおよび閉じるタグ
<last>Smith</last>      // 大文字小文字が一致
</name>                 // ルート要素の閉じるタグ
```

この例題では<name>がルート要素で、<first>と<last>が子要素です。また<first>と,last>はお互いに兄弟要素です。

XML は HTML の拡張やアップグレードではありません。それらはそれぞれ異なる目的を持っています。HTML は Web サイトのデータを表示し、その主たる目的はデータがどのように見えるかです。他方 XML はデータの格納と転送の他には何も行わず、データそのものにフォーカスがあります。XML はテキストファイルで、データを読み込むアプリケーションが解析を行います。XML ファイルを表示するには、eXtended Style Language (XSL) の利用が必要です。

XML 標準は W3C により推奨されています。これらの標準によると、XML ドキュメントは*整形式*または*妥当*でなければなりません。整形式なドキュメントはシンタックスが正しいドキュメントであり、妥当なドキュメントは整形式かつ *Document Type Definition (DTD)* のルールに適合しているものです。

Document Type Definition (DTD) は利用できる要素構造や属性および要素タグの内側に位置する値を定義します。DTD は埋め込みあるいは外部ファイル (.dtd) への参照として定義されます。

4D における XML

XML をサポートするために、4D は Apache Foundation の Xerces.dll と Xalan_C_1_6_0.dll を使用します。

*Xerces*はXMLを解析し操作するためのライブラリです。ライブラリはDOM、SAX、SAX2 を含む多数の標準を実装します。詳細は http://xerces.apache.org/ を参照してください。

*Xalan*はオープンソースライブラリで、XSLTやXPathランゲージを実装します。詳細は http://xalan.apache.org/ を参照してください。

既存のスタイルシートを使用して XML ドキュメントを変換する 4D コマンドがあります:

APPLY XSLT TRANSFORMATION (xmlSource; xslSheet; result{; compileSheet})

このコマンドは XML 構造を含む BLOB またはドキュメントに XML 変換を適用し、ドキュメントまたは BLOB を生成します。引数は:

xmlSource	XML ソースが格納されたドキュメントのパスまたは BLOB。
xslSheet	XSL スタイルシートが格納されたドキュメントのパスまたは BLOB。
result	XSLT 変換の結果を受け取るドキュメントパスまたは BLOB。
compileSheet	オプション。この引数が True に設定されると、XML ソースファイルが最初の呼び出し時に解析され、コンパイル、そしてメモリに格納されます。

4D では XSL スタイルシートのパラメタを以下のコマンドで変更できます:

SET XSLT PARAMETER (paramName; paramValue)

paramName	XSL スタイルシート中で探すパラメタ名
paramValue	変換されるドキュメントで使用されるパラメタの値

両コマンドは一緒に使用され、同じプロセス内で実行します。

DOM と SAX の違い

4D の XML 解析には 2 つのモードがあります。

DOM (Document Object Model) は XML 構造をメモリ上に構築します。それぞれの要素へのアクセスは高速です。他方大きな XML 構造の解析時にはメモリを大きく消費します。

SAX に必要なメモリの量は、XML ファイルの最大深度と 1 つの要素の XML 属性に格納された最大データによります。言い換えれば SAX モードでは 1 つの要素、そのサブ要素、そしてそのデータだけがメモリに格納されます。この要素構造が解析されるとき、次の要素構造がロードされます。要素構造はツリー構造全体より小さいものです。言い換えればサイズが大きな XML ドキュメントを扱う際には、利用可能なメモリ量にかかわらず SAX モデルをお勧めします。

XPath 記法

XPath 記法は XPath ランゲージから来ています。主たる目的は XML ドキュメントの特定の部分にアクセスすることです。また文字列や数値、ブールを扱う基本的な機能も提供します。

element3 にアクセスする XPath 記法の例:

```
<root>
   <element1>
          <element2>
                 <element3>
                 </element3>
          </element2>
   </element1>
</root>
```

element3 にアクセスする XPath 記法は:

```
/root/element1/element2/element3
```

4D は XPath 参照スタイルを **DOM Create XML element**、**DOM Find XML element**、**DOM SET XML ELEMENT VALUE** コマンドで使用します。この方法により 4D でプログラムにより要素を簡単に作成、アクセス、そして操作できます。

4D の SAX コマンドは XML をディスクに作成、処理、保存します。これらのコマンドはドキュメント参照を使用し、またドキュメントテーマのコマンド **OPEN DOCUMENT**、**APPEND DOCUMENT**、**SEND PACKET** などと共に使用できます。

例題

ここでの例題では、DOM と SAX コマンドを使用して、同じ XML ファイルを作成します。

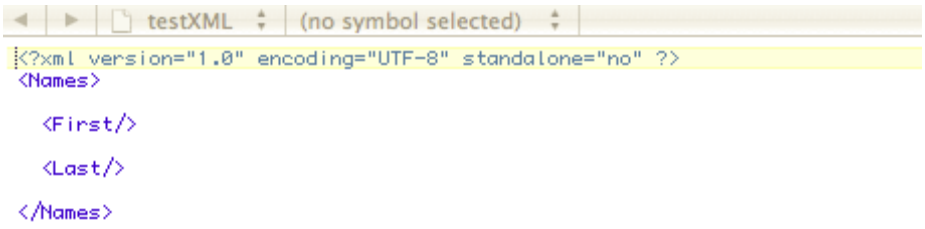
DOM テーマのコマンドを使用する

このコードはアクティブな 4D フォルダ内に"testXML.xml"という名前の XML 構造ファイルを作成します。

```
C_STRING(16;Ref)

Ref:=DOM Create XML Ref(“Names”)              ` ルート参照を作成
DOM Create XML element(Ref;”/Names/First”)  ` First 子要素を作成
DOM Create XML element(Ref;”/Names/Last”)   ` First の兄弟要素を作成
DOM EXPORT TO FILE(Ref;”testXML.xml”)       ` ディスクに XML を保存
DOM CLOSE XML(Ref)                            ` メモリを解放
```

以下は作成されたファイルです:



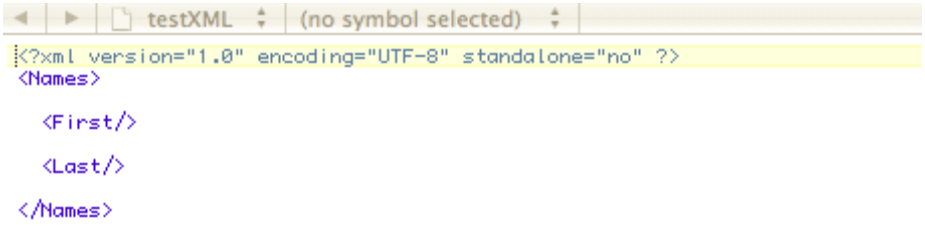
SAX テーマのコマンドを使用する

このコードはアクティブな 4D フォルダ内に”testXML.xml”という名前の XML 構造ファイルを作成します。

```
C_TIME(docRef)

docRef:=Create document(“TestSAX.xml”)      ` ドキュメントを作成
SAX OPEN XML ELEMENT(docRef;”Names”)        ` ルート要素を作成
SAX OPEN XML ELEMENT(docRef;”First”)        ` 子ノードを作成
SAX CLOSE XML ELEMENT(docRef)               ` 子ノードを閉じる
SAX OPEN XML ELEMENT(docRef;”Last”)         ` “First”ノードの兄弟要素を作成
SAX CLOSE XML ELEMENT(docRef)               ` 子ノード“Last”を閉じる
CLOSE DOCUMENT(docRef)                      ` ドキュメントを閉じる
```

以下は作成されたファイルです:



両ファイルは同じ結果となります。

Note: *XML エラーは 9911 から 9935 のコードが返され、その説明はランゲージリファレンスにあります。*

http://www.4d-japan.com/docs/CMJ/CMJ02024.HTM

4D DTD フォルダ

4D DTD フォルダは 4D ソフトウェアの”Resources”フォルダに配置されます。Mac OS X では”Resources”フォルダは 4D パッケージの中にあります。Windows では 4D アプリケーションフォルダの中です。DTD フォルダには 4D が XML 構造を検証するために必要となるすべてのシンタックスルールが格納されています。

XML 構造はテーブルのコピーや新しい 4D データベースを開く際には検証される必要はありません。言い換えれば、以下の行を構造から取り除くことができ、4D はこのドキュメントを内部的に検証します。

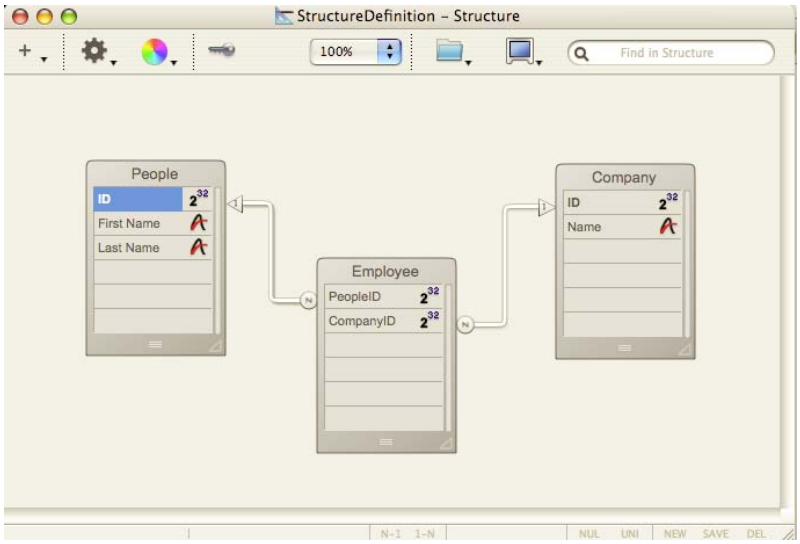
```
<!DOCTYPE base SYSTEM “http://www.4d.com/dtd/2007/base.dtd>
```

XML 構造が整形式でなく、DTD フォルダ内の”base_core.dtd”や“common.dtd”ファイルで定義されたシンタックスルールに準じていない場合、エラーが生成されます。4D はテーブルなしの新しく作成されたデータベースを開きます。

4D v11 ストラクチャ定義

4D ストラクチャ定義は、4D ストラクチャエディタウィンドウで作成できるすべてのオブジェクト、テーブル、フィールド、インデックス、リレーション、そしてストラクチャエディタ設定を含んでいます。

例:



この例には以下のオブジェクトが含まれています:

- 3 テーブル (People、Company、Employee)
- 2 リレーション (Link_1、Link_2)
- フィールドタイプ (倍長整数、文字タイプ)
- インデックス ([People]ID と[Company]ID に自動インデックスを設定

これらすべてのオブジェクトとプロパティはストラクチャウィンドウで作成されています。

ストラクチャ定義の使用方法

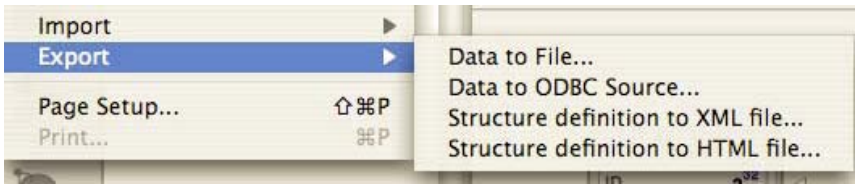
先に、4D ストラクチャ定義を使用して新規データベースを作成したり、4D データベース間でストラクチャオブジェクトを移動できると言いました。

Note: 4D と XML をサポートする他のアプリケーション間でストラクチャオブジェクトを移動するには、XML 構造がシンタックスルールに沿っており、そのストラクチャ定義を使用するアプリケーション用に定義された XSLT スタイルシートにより整形式でなければなりません。後にこのアプリケーションは、DTD で定義されたランゲージルールに基づき、ストラクチャ定義の検証を試みます。

この節ではこれらの処理を行う方法と、4D が要素タグ内でストラクチャエディタ設定を管理する方法を詳細に説明します。

ストラクチャ定義の書き出し

4D v11 SQL では XML ファイルへの書き出し機能を使用して、ストラクチャ構造を書き出すことができます。これはデザインモードの"ファイル"メニューから可能です。



保存ダイアログボックスで XML ファイルの名前と保存先を選択できます。

以下は XML ドキュメントに書き出されたデータベースストラクチャの例です:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE base SYSTEM "http://www.4d.com/dtd/2007/base.dtd" >
<base name="StructureDefinition" uuid="9B4C8D6FD59A4918B7F6DEC2799F85DF"
collation_locale="en">
    <table name="People" uuid="31B158A370724E0187AA0D46981CC90A">
          <field name="ID" uuid="5A646F2A697B4703A0E7F956AE6FFC2A" type="4"
unique="true" never_null="true">
                <index_ref uuid="92F6B07050B946B09F635770000AD527"/>
                <field_extra mandatory="true"/>
          </field>
          <field name="First Name" uuid="C8E8B9F5E96F4BA5AFFFFD3C1A1E71F8"
type="10" limiting_length="80" never_null="true"/>
          <field name="Last Name" uuid="AC19ABE1445D4174994206C0671B760E"
type="10" limiting_length="80" never_null="true"/>
          <table_extra>
                <editor_table_info>
                      <color red="255" green="255" blue="255" alpha="0"/>
```

```
                        <coordinates left="39" top="40" width="120"
height="168"/>
                </editor_table_info>
          </table_extra>
    </table>
    <table name="Company" uuid="357C9E3C506D430FA47659EA7FBE402F">
          <field name="ID" uuid="854616B40C0C4D17A9716CAC0201E153" type="4"
unique="true" never_null="true">
                <index_ref uuid="095B2E5A58814501BF8CF72FC2779236"/>
                <field_extra mandatory="true"/>
          </field>
          <field name="Name" uuid="86DC6F07FF744F85AF338003E907B6C6" type="10"
limiting_length="100" never_null="true"/>
          <table_extra>
                <editor_table_info>
                      <color red="255" green="255" blue="255" alpha="0"/>
                      <coordinates left="444" top="43" width="120"
height="168"/>
                </editor_table_info>
          </table_extra>
    </table>
    <table name="Employee" uuid="C0EBFEA32D40456AB9A28C37E0422FFE">
          <field name="PeopleID" uuid="02634015D43A4257821327C866DB07D6"
type="4" never_null="true"/>
          <field name="CompanyID" uuid="934220AB9B644F28B93A75A219D3F749"
type="4" never_null="true"/>
          <table_extra>
                <editor_table_info>
                      <color red="255" green="255" blue="255" alpha="0"/>
                      <coordinates left="238" top="124" width="120"
height="168"/>
                </editor_table_info>
          </table_extra>
    </table>
    <relation uuid="ABFA4A957C8B4AA6B52CD25069A86CAE" name_Nto1="Link_1"
name_1toN="Link_1_return" auto_load_Nto1="true" auto_load_1toN="true"
foreign_key="false" state="1">
          <related_field kind="source">
                <field_ref uuid="934220AB9B644F28B93A75A219D3F749"
name="CompanyID">
                      <table_ref uuid="C0EBFEA32D40456AB9A28C37E0422FFE"
name="Emploee"/>
                </field_ref>
          </related_field>
          <related_field kind="destination">
                <field_ref uuid="854616B40C0C4D17A9716CAC0201E153" name="ID">
                      <table_ref uuid="357C9E3C506D430FA47659EA7FBE402F"
name="Company"/>
                </field_ref>
          </related_field>
          <relation_extra entry_wildchar="false" entry_create="false"
choice_field="0" entry_autofill="false">
                <editor_relation_info via_point_x="0" via_point_y="0"
prefers_left="false" smartlink="true">
                      <color red="255" green="255" blue="255" alpha="0"/>
                </editor_relation_info>
          </relation_extra>
    </relation>
    <relation uuid="341AA380539344E2BF5EEBC2FEB39742" name_Nto1="Link_2"
name_1toN="Link_2_return" auto_load_Nto1="true" auto_load_1toN="true"
foreign_key="false" state="1">
          <related_field kind="source">
                <field_ref uuid="02634015D43A4257821327C866DB07D6"
name="PeopleID">
                      <table_ref uuid="C0EBFEA32D40456AB9A28C37E0422FFE"
name="Emploee"/>
                </field_ref>
          </related_field>
          <related_field kind="destination">
```

```
                  <field_ref uuid="5A646F2A697B4703A0E7F956AE6FFC2A" name="ID">
                        <table_ref uuid="31B158A370724E0187AA0D46981CC90A"
name="People"/>
                  </field_ref>
            </related_field>
            <relation_extra entry_wildchar="false" entry_create="false"
choice_field="0" entry_autofill="false">
                  <editor_relation_info via_point_x="0" via_point_y="0"
prefers_left="true" smartlink="true">
                        <color red="255" green="255" blue="255" alpha="0"/>
                  </editor_relation_info>
            </relation_extra>
      </relation>
      <index kind="regular" unique_keys="true"
uuid="095B2E5A58814501BF8CF72FC2779236" type="7">
            <field_ref uuid="854616B40C0C4D17A9716CAC0201E153" name="ID">
                  <table_ref uuid="357C9E3C506D430FA47659EA7FBE402F"
name="Company"/>
            </field_ref>
      </index>
      <index kind="regular" unique_keys="true"
uuid="92F6B07050B946B09F635770000AD527" type="7">
            <field_ref uuid="5A646F2A697B4703A0E7F956AE6FFC2A" name="ID">
                  <table_ref uuid="31B158A370724E0187AA0D46981CC90A"
name="People"/>
            </field_ref>
      </index>
</base>
```

ルート要素は 7 つの子要素 (3 テーブル、2 リレーション、2 インデックス) を持っています。

この節ではストラクチャ定義ファイルに現れる各要素を説明します。

base

これはストラクチャ定義のルート要素です。XML はカスタマイズされたタグで動作するので、このタグは XML のシンタックスルールに違反しません。

table

これはルート要素の子要素です。すべてのテーブルはルート要素内の兄弟要素です。属性は:

- name - ストラクチャエディタに設定されたテーブル名。
- uuid - 4D が自動で設定するユニーク ID。開発者がこの番号をコントロールしたり変更したりすることはできません。XML ドキュメント内では、要素参照番号として使用されます。

field

これは table 要素の子要素です。<field>要素の数はテーブル中のフィールド数と一致します。すべての<field>要素は兄弟要素です。言い換えれば field 要素をネストすることはできません。属性は:

- name - ストラクチャエディタで設定されたフィールドの名前。
- uuid - フィールド参照番号
- type - フィールドタイプ:
    - 1 - ブール
    - 2 - 倍長整数
    - 3 - 整数
    - 5 - Integer 64 bits
    - 6 - 実数
    - 7 - Float
    - 8 - 日付
    - 10 - 文字
    - 18 - BLOB
    - 20 - ピクチャ
- limited_length - この属性は文字フィールドに設定され、文字の上限値を示します。
- unique - この属性はインスペクタウィンドウで重複不可オプションにチェックが入っているときにのみ、field タグ内に表示されます。このパラメタの値は"true"に設定されます。
- never_null - この属性はインスペクタウィンドウで Null 値を空値にマップがチェックされていると有効になります。

index_ref

これはインデックスが設定されたフィールドで有効です。このフィールドの唯一の属性は uuid 番号です。この要素は field 要素の子要素です。

field_extra

これは index_ref の兄弟要素で、属性はインスペクタウィンドウのデータ入力制御エリアで設定されるものです。

tip

この要素は field_extra の子要素で、Help Tip ボックスに入力されたテキストを含みます。

table_extra

これは table 要素の子要素です。この要素は<comment>と<editor_table_info>要素を含みます。comment タグはコメントウィンドウのスタイル定義とコメントテキストを含みます。

editor_table_info

この要素はストラクチャエディタウィンドウ内でのテーブルのカラーや座標に関する情報を含みます。

relation

これはルート要素の子要素です。

- uuid - 参照番号。
- name_Nto1 と name_1toN - リレーション名。
- auto_load_Nto1 と auto_load_1toN - このリレーションに設定された自動 N 対 1 リレートと自動 1 対 N リレートの値に対応する"true"または"false"。
- stage - リレーションが設定されていれば"1"、リレーションを無効にするには"0"。

related_field

このタグには"kind"属性のみがあります。この属性は 2 つの値"source"または"destination"のいずれかをとります。

field_ref と table_ref

この両タグはフィールドとテーブルの参照を属性として含んでいます。これらはフィールド uuid とテーブル uuid 番号です。

relation_extra

これには自動ワイルドカードサポートに関する情報が含まれます。

editor_relation_info

このタグにはリレーションの表示に関する情報を含みます。

index

この要素はルート要素の子要素です。属性は:

- unique_keys - インスペクタウィンドウで重複不可ボックスがチェックされているかいないかで true または false が設定されます。

- uuid - インデックス参照番号
- type - 子要素<field_ref>と<table_ref>の参照番号を持つフィールドに設定されたインデックスのタイプまたインデックスに名前が設定されていればその name 属性。

base_extra

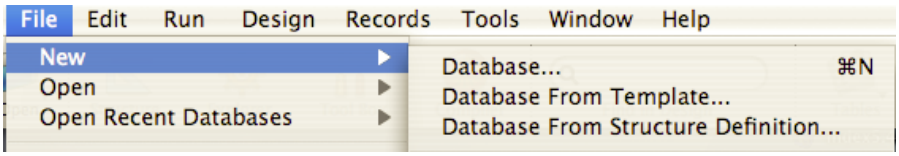
このタグはパッケージ名、ストラクチャファイル名、データファイル名を含みます。

journal_file

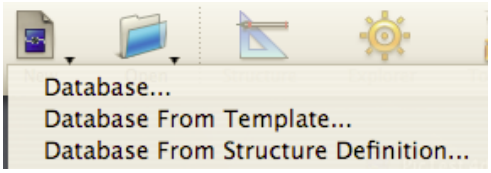
この要素は base_extra の子要素で、環境設定のバックアップページでログファイルチェックボックスがチェックされているときにのみ含まれます。

ストラクチャ定義の読み込み

ストラクチャ定義から新規データベースを作成できます。
4D を開き、ファイルメニューから新規 > ストラクチャ定義からデータベースを選択します。



またはツールバーの新規アイコンからも選択できます。



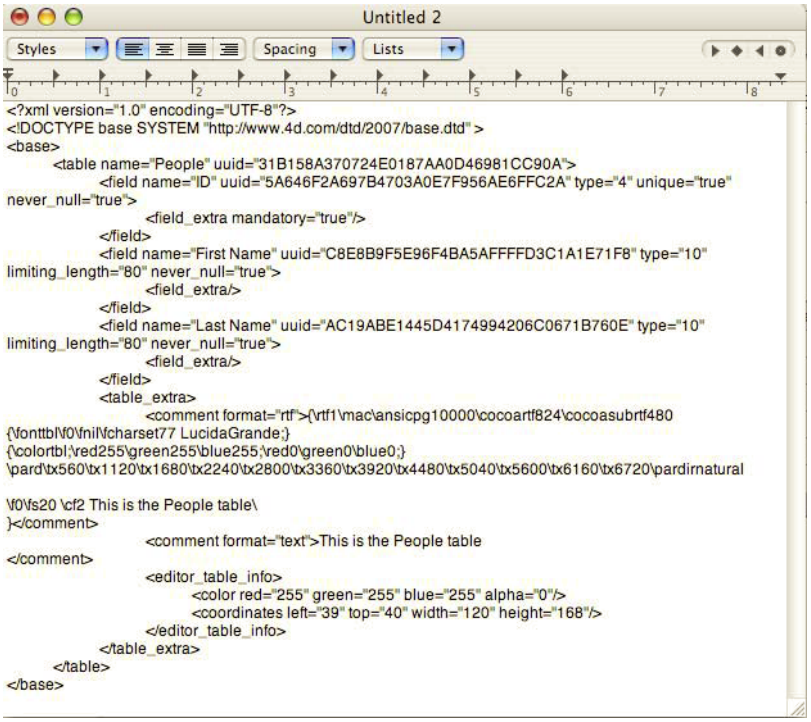
ファイルを開くダイアログウィンドウが表示され、XML ファイルを指定できます。

Note: ストラクチャ定義ファイルの場所はローカルディスクである必要はありません。他のドライブやネットワークパーティションのファイルも指定できます。

ストラクチャオブジェクトのコピー

ストラクチャエディタで編集メニューのコピー/ペーストコマンド、あるいはコンテキストメニューを使用して、オブジェクトをコピーできます。コピーを行うと 4D はクリップボードにコピーされたオブジェクトの XML 構造を配置します。

以下はテキストエディタにペーストしたテーブル XML 構造です。



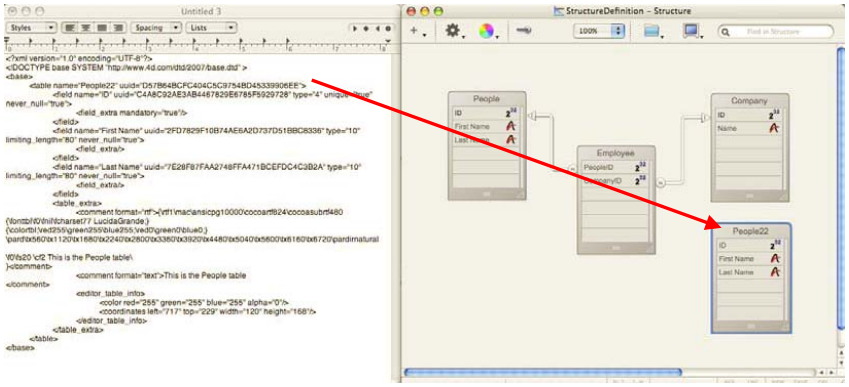
上記の例では[People]テーブルをコピーし、TextEdit にペーストしました。

この方法ではテーブル、フィールド、インデックス、リンクプロパティを直接テキストエディタで編集できます。またテーブルテンプレートを作成して、新規データベースや既に開かれたデータベースでそれを使用できます。

UUID (Universal Unique Identifier) は、その名前が示す通り、4D がすべての 4D オブジェクトに割り当てるユニークな番号です。この番号は XML 構造内で要素参照番号としても使用されます。

XML 構造をストラクチャエディタにペーストすると、4D は異なる UUID 番号を新しく作成されたオブジェクトに割り当てます。

次の例題ではテキストエディタで、テーブル名を[People22]に変更し、XML をコピーして、ストラクチャエディタにペーストしています。新たに作成されたテーブル[People22]は異なるテーブル番号と UUID を持ちます。フィールドの UUID も異なります。



まとめ

このテクニカルノートではストラクチャ定義ファイルを作成する方法と、このファイルから新しいデータベースを作成する方法を示しました。ストラクチャ定義ファイルは XML 形式であり、XML 言語のポテンシャルを完全に使用することができます。

4D v11 SQL でサポートされたストラクチャ絵意義について説明する前に、XML 言語について概観しました。完全に説明したわけではありませんが、ストラクチャ定義ファイルのシンタックスと構造を理解するために最小限の説明を行いました。

この 4D の新しい機能は 4D デベロッパに、古いストラクチャからテーブルやフィールドを、その属性やプロパティとともにコピーして新しいデータベースストラクチャを作成するハンディなツールを提供します。