Zen v15

DDF Builder User's Guide

Guide to Using DDF Builder

ACTIAN™

DDF Builder User's Guide
2021 年 10 月

目次

このドキュメントについて

このドキュメントは DDF Builder で作業を行うためのガイドです。DDF Builder の概要、グラフィカル ユーザー インターフェイスを構成するコンポーネントについての説明、およびこのユーティリティでさまざまな作業を行う際に役立つチュートリアルが含まれています。

このドキュメントの読者

このドキュメントは、DDF Builder を使用するユーザーを対象としています。このドキュメントで説明している手順を実行するには、データの構造に関するある程度の知識やトランザクショナル アクセス方法とリレーショナル アクセス方法の基本的な知識が必要です。

必要に応じて、知識が必要であると思われる特定の概念や方法についてはこのドキュメント内で明確に提示しています。それら提示された事項について理解していない場合は、その知識を得てから DDF Builder を使用するようにしてください。

表記上の規則

特段の記述がない限り、コマンド構文、コード、およびコード例では、以下の表記が使用されます。

大文字と小文字の区別	通常、コマンドと予約語は、大文字で表記されます。本書で別途記述がない限り、これらの項目は大文字、小文字、あるいはその両方を使って入力できます。たとえば、`MYPROG`、`myprog`、または `MYprog` と入力することができます。
太字	太字で表示される単語には次のようなものがあります。メニュー名、ダイアログ ボックス名、コマンド、オプション、ボタン、ステートメントなど。
固定幅フォント	固定幅フォントは、コマンド構文など、ユーザーが入力するテキストに使われます。
[ ]	省略可能な情報には、［*log_name*］のように、角かっこが使用されます。角かっこで囲まれていない情報は必ず指定する必要があります。
|	縦棒は、［*file name* ｜ *@file name*］のように、入力する情報の選択肢を表します。
< >	＜＞ は、`/D=<5|6|7>` のように、必須項目に対する選択肢を表します。
変数	*file name* のように斜体で表されている語は、適切な値に置き換える必要のある変数です。
...	［*parameter...*］のように、情報の後に省略記号が続く場合は、その情報を繰り返し使用できます。
::=	記号 ::= は、ある項目が別の項目用語で定義されていることを意味します。たとえば、a::=b は、項目 *a* が *b* で定義されていることを意味します。

DDF Builder について

1

Btrieve ファイルへのリレーショナル アクセスの提供

DDF Builder の概要

DDF Builder は Java ユーティリティで、基となるデータ ファイル（以後、「Btrieve ファイル」と言います）に変更を加えることなく、Zen データ辞書ファイル（DDF）の表示、作成、変更が行えます。DDF Builder ではさまざまな機能を提供しますが、主に以下のような目的で使用できます。

- 既存の Btrieve データ ファイルへのリレーショナル アクセスを可能にするために必要なテーブル定義を作成する
- Btrieve データ ファイルへのリレーショナル アクセスを確実に正しく行えるよう既存のテーブル定義を変更する

メモ DDF Builder で DDF を作成したり変更しても、Btrieve データ ファイルが変更されることはありません。

DDF Builder には上記以外にも次のような機能があります。また、これ以外にもさらに機能があります。

- MicroKernel エンジン で使用する Btrieve データ ファイルを新規作成する
- Btrieve スキーマを XML ファイルへエクスポートする
- XML ファイルから Btrieve スキーマへインポートする
- SQL ステートメントを作成する

DDF Builder を使用する理由

DDF Builder は、Btrieve データ ファイルを変更することなく、Btrieve データ ファイルへのリレーショナル アクセスを追加することができる特殊なユーティリティです。通常、DDF Builder は日常的に使用するユーティリティではありません。DDF Builder は既存の Btrieve ファイルへのリレーショナル アクセスを追加するために必要に応じて使用するもので、データ辞書ファイルのテーブル定義を作成したり、正しく構成されていないテーブル定義へ接続するために既存のデータ辞書ファイルを変更したりします。

メモ DDF は SQL メタデータ用のスキーマを定義します。DDF は、DDF Builder で SQL アクセスをオブジェクト、つまり SQL テーブルで表すことができるようにするシステム テーブルです。DDF ディレクトリを変更するのではなく、SQL テーブル オブジェクトに対して作業します。DDF Builder では、SQL テーブルを作成、変更または削除した場合に DDF を変更します。

1 つのデータベース内にあるすべての SQL テーブルは同じ DDF のセットに定義されます。

旧バージョンの Zen Control Center のテーブル設計では、**リンク モード**と**非リンク モード**の 2 つの作業モードを提供しているものもあります。リンク モードではテーブル定義とデータ ファイルの両方を変更することができますが、非リンク モードで変更できるのはテーブル定義のみです。DDF Builder では、IN DICTIONARY 呼び出しを使って非リンク モード同様のモードを使用することで、Btrieve ファイルを変更することなく DDF を変更します。DDF Builder では Btrieve ファイルへ書き込みを行うことはありません。

DDF Builder では行わないこと

DDF Builder は既存の Btrieve データ ファイルを変更する目的には使用できません。既存の Btrieve ファイルおよび DDF を変更してリレーショナル アクセスを提供するつもりであれば DDF Builder を使用すべきではありません。その場合は、既存の Btrieve ファイルを変更する Zen Control Center やその他の Zen ユーティリティを使用することをご検討ください。

また、DDF Builder はキーの作成や変更に使用するものではありません。キーに関する操作を行う場合は、Zen Control Center の Table Editor を使用します DDF Builder は DDF の作成や変更を目的としています。

ZenCC を使用しない理由

Zen Control Center は物理データ ファイルとデータ辞書ファイルを同時に操作する、つまりリンク モードで操作することを目的としています。DDF のみを変更するために ZenCC を使用することはお勧めしません。

必要な知識

DDF Builder を使用する場合に最も重要なのは、作業対象のデータの構造を完全に理解しておくことです。データの構造を理解していないと、DDF Builder でテーブル定義を作成したり変更することが難しくなるため時間がかかり、このユーティリティの効果が得られません。

その他の有用な情報

Zen データベースからデータがアクセスされるときに用いられるトランザクショナルとリレーショナルという 2 種類の主要なアクセス方法について理解していることも役立ちます。DDF Builder の機能の大部分は、トランザクショナル アクセスについての知識が役立ちます。それ以外の機能については、リレーショナル データベースの一般的な概念についてよく理解していることが必要です。

メモ このマニュアルは、トランザクショナル アクセス方法とリレーショナルの概念について理解していることを前提として説明しています。これらトランザクショナルおよびリレーショナルに関する説明や、キー、インデックス、ページ タイプ、スキーマなどの一般的な用語についての定義は行いません。

DDF Builder を使用する前に理解を深めておく必要がある場合は、下記の Zen マニュアルを参照してください。デフォルトで、Zen 開発者向けドキュメントは Zen データベース エンジンのインストール時に一緒にインストールされます。

トランザクショナル アクセス

トランザクショナル アクセスの場合、アプリケーション プログラムは物理的または論理的のどちらの経路でもデータ レコード内を前方および後方に移動することができます。トランザクショナル API を使用することで、アプリケーション プログラムは直接制御を備え、開発者はデータの基本構造の知識に基づいてデータ アクセスを最適化することができます。Btrieve API を使用することはトランザクショナル アクセスの 1 例です。

トランザクショナル アクセスについて理解を深めるには、以下のマニュアルを参照してください。

- *Zen Programmer's Guide*（開発者リファレンス）
- *Btrieve API Guide*（開発者リファレンス）
- *Advanced Operations Guide*（上級者リファレンス）

リレーショナル アクセス

リレーショナルとは、データがテーブル、行、列の集まりとして表されるアクセス方法です。リレーショナル モデルは、開発者を基となるデータ構造から切り離し、データを単純な表形式で表します。ODBC はリレーショナル アクセス方法の 1 例です。

リレーショナル アクセスについて理解を深めるには、以下のマニュアルを参照してください。

- *SQL Engine Reference*（上級者リファレンス）
- *Advanced Operations Guide*（上級者リファレンス）
- *Zen User's Guide*（一般リファレンス）
- *Zen Programmer's Guide*（開発者リファレンス）

次に行うこと

DDF Builder でテーブル定義に対して変更を行うと DDF の構造が変更されます。このユーティリティの使用前の注意事項として、作業対象のファイル（DDF も含む）を必ずバックアップしておいてください（Btrieve ファイルは、ファイルのページ構造内にデータが格納されるため、データ ファイルとも呼ばれます）。

メモ DDF Builder では既存の Btrieve ファイルのレコード レイアウト構造を変更できません。このユーティリティを使って新しい Btrieve ファイルを作成することはできます。

セキュリティの無効化

Zen セキュリティ モデルが有効なデータベースに対して作業する場合、DDF Builder でそのファイルを開く前にはそのデータベースをオフラインにし、すべてのセキュリティを解除しておきます。

DDF Builder のコンポーネント

ユーティリティ以外にも、DDF Builder では以下のコンポーネントがあります。

ログ ファイル

ログ ファイルは DDF Builder が問題のある状態を報告するファイルです。デフォルトのインストールで、ログ ファイルは次のディレクトリにインストールされます。

C:¥ProgramData¥Actian¥Zen¥rcp¥workspace-builder¥.metadata

ユーザー向けドキュメント

『*DDF Builder User's Guide*』は Zen ユーザー向けドキュメントに含まれています。

チュートリアル ファイル

DDF Builder では、『DDF Builder』に含まれているチュートリアルで使用するファイルのセットをインストールします。このチュートリアル ファイルはデフォルトで、C:¥ProgramData¥Actian¥Zen¥DDFBuilder¥tutorials にインストールされます。説明については、「DDF Builder チュートリアル」を参照してください。

デフォルトのインストール先については『*Getting Started with Zen*』の「ファイルはどこにインストールされますか？」を参照してください。

DDF Builder の起動

DDF Builder の起動は、Windows オペレーティング システムから、Zen Control Center（ZenCC）内から、あるいはコマンド ラインから行うことができます。

▶▶ DDF Builder を Windows オペレーティング システムから起動するには

1 ［**スタート**］メニューまたは**アプリ**画面から DDF Builder にアクセスします。

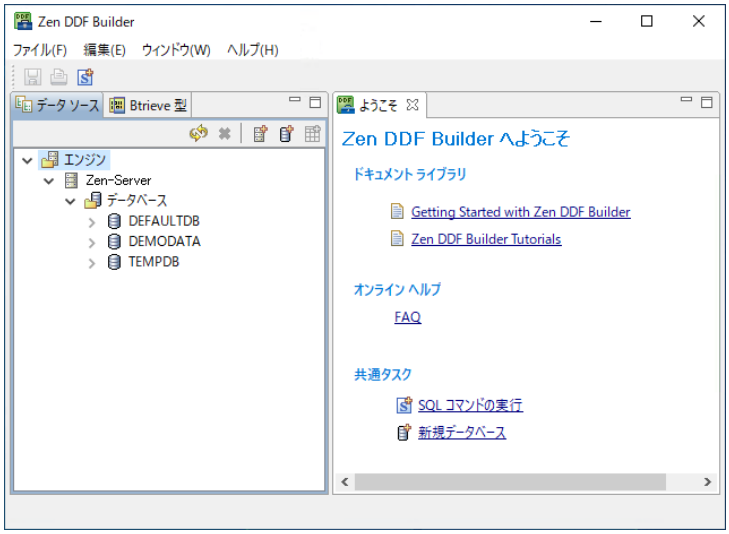
DDF Builder メイン ウィンドウが開きます。

▶▶ ZenCC から DDF Builder を起動するには

1 ZenCC のメニュー バーで［**ツール**］>［DDF Builder］をクリックします。

2 DDF Builder メイン ウィンドウが開きます。

図 1 DDF Builder メイン ウィンドウ



コマンド ラインから DDF Builder を起動する

DDF Builder は Windows、Linux または macOS でコマンド ラインから実行可能ファイルを実行することによって起動することができます。詳細については、次のトピックを参照してください。

Windows

実行可能ファイル builder.exe を実行すると、DDF Builder が起動します。『*Getting Started with Zen*』の「ファイルはどこにインストールされますか？」を参照してください。

Linux および macOS

実行可能スクリプト ファイル builder を実行すると、DDF Builder が起動します。このスクリプト ファイルは、デフォルトで usr/local/actianzen/bin ディレクトリにインストールされます。

ファイル ブラウザー アプリケーションでスクリプト ファイルをダブルクリックするのではなく、コマンド プロンプトから DDF Builder を起動することをお勧めします。「DDF Builder の実行に関するトラブルシューティング ガイド」を参照してください。

Linux または macOS 上で DDF Builder を起動するには、以下の要件を満たしている必要があります。

表 1　Linux または macOS で DDF Builder を起動するための要件

要件	説明
Zen Server または Client	同一マシンに互換性のある Zen Server または Client が既にインストールされている必要があります。『*Getting Started with Zen*』の「Zen（Linux ベースのシステム用）のインストール」を参照してください。
X Server へのアクセス	xhost コマンドは、どのクライアントが現在のマシンの X-Windows System にアクセスできるかを制御します。デフォルトで、xhost はアクセス制御をオンにします。つまり、X-Windows System を起動したユーザーのみが DDF Builder を起動できます。 X-Windows System クライアントの制限は、ターミナル ウィンドウで `xhost +` を実行することによって解除できます。
Java Runtime Environment（JRE）	DDF Builder を実行するために必要な JRE コンポーネントは、Zen の一部としてインストールされます。DDF Builder は、Zen の一部としてインストールされる JRE のローカル バージョンを使用します。
デスクトップの所有権（macOS のみ）	デスクトップにログインしているユーザーのみが DDF Builder を起動することができます。

DDF Builder を実行する要件に合致しているのに、実行に問題がある場合は、以下のトラブルシューティング ガイドを参考にしてください。

表 2　DDF Builder の実行に関するトラブルシューティング ガイド

トラブルシューティングする状態	説明
"java.lang.UnsatisfiedLinkError" というエラーを受け取った。	このエラーは、ファイル ブラウザー アプリケーションを使用してスクリプト ファイルをダブルクリックして DDF Builder を起動しようとしたときに、よく起こります。コマンド プロンプトから DDF Builder を起動してください。 このエラーは、LD_LIBRARY_PATH 変数が設定されていない場合に発生します。builder スクリプトがこの変数を設定します。この変数は、以下のコマンドを使用して明示的に設定することもできます。 `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/actianzen/lib64` macOS では、この変数は DYLD_LIBRARY_PATH になります。

DDF Builder を root またはユーザー zen-svc として実行しようとすると、"SWT no more handles" というエラーが返される。	DDF Builder を実行するには、zen-svc または root ユーザーとしてログインする必要はありません。ただし、これらのユーザーのどちらでもない場合は、zen-data グループのメンバーである必要があります。
	"SWT no more handles" エラーは、X Server がクライアントへの接続を拒否したために発生します。zen-svc または root ユーザーに切り替える前に、コンソール ウィンドウを開き、`xhost +` と入力してほかのクライアントが X Server に接続できるようにします。
	これで、zen-svc または root ユーザーに切り替えることができます。
	また、場合によっては DISPLAY 環境変数を設定する必要があります。zen-svc または root ユーザーとして、コンソール ウィンドウで以下のコマンドを入力します。
	`export DISPLAY=:0.0`
	または
	`export DISPLAY=localhost:0.0`

表 2　DDF Builder の実行に関するトラブルシューティング ガイド

トラブルシューティングする状態	説明
DDF Builder のエラー ログ ファイルを表示、またはエラーをコンソール ウィンドウにリダイレクトしたい。	DDF Builder エラーのログ ファイルは、ユーザーのホーム ディレクトリ下にあります。トラブルシューティングのためには、エラーをコンソール ウィンドウにリダイレクトするとより便利です。 エラーをコンソール ウィンドウにリダイレクトするには、DDF Builder の起動時に -consoleLog オプションを使用します。 `builder -consoleLog`
以下のエラー メッセージが返された。" データベース エンジンに接続できません。ターゲット マシンがアクセス可能で、かつエンジンが実行されていることを確認してください。 "	このエラー状況は、ローカル サーバーを管理しようとした場合に発生します。 ローカル サーバーを管理するには、zen-data グループのメンバーであるか root ユーザーである必要があります。『*Getting Started with Zen*』の 「Linux、macOS、Raspbian での Zen のアカウント管理」を参照してください。
root 以外のユーザーとして DDF Builder を起動しようとすると、"GTK IM Module SCIM: Cannot connect to Panel!" というエラーが返される。	いくつかの Linux OS では、環境変数 GTK_IM_MODULE を指定する必要があります。 この問題を解決するには、DDF Builder を起動する前にコンソール ウィンドウで、以下のコマンドを入力します。 `export GTK_IM_MODULE=scim-bridge`

DDF Builder のキャッシュをクリアする

DDF Builder は効率を上げるために一定の情報をキャッシュします。弊社テクニカル サポートからの指示に応じてトラブルシューティング目的に、あるいはすべてのファイルを確実に再ロードするために、キャッシュをクリアする必要がある場合もあります。キャッシュは、DDF Builder をパラメーターを使ってコマンド ラインから起動した場合にのみクリアできます。

メモ DDF Builder の通常の使用時には、起動時にキャッシュをクリアしても何も利点はありません。通常使用の場合は、「DDF Builder の起動」の説明のとおりにユーティリティを起動します。

▶▶ DDF Builder のキャッシュをクリアするには

1　DDF Builder を実行中の場合は終了します。

2　コマンド プロンプトを開きます。

3　Zen インストール ディレクトリの Zen¥bin¥ ディレクトリに移動します。

4　「`builder -clean`」と入力して、Enter キーを押します。

これにより、キャッシュがクリアされ、DDF Builder が起動します。

次に行うこと

DDF Builder の全般的な理解が得られたら、「DDF Builder の使用」でユーザー インターフェイスの概要を参照してください。

DDF Builder の使用

2

DDF Builder での作業ガイド

この章では、以下の項目について説明します。

- 「DDF Builder の概念」
- 「GUI のリファレンス」
- 「DDF Builder の各種作業」

DDF Builder の概念

このセクションでは、DDF Builder での作業に関する概念的な情報を提供します。以下のセクションがあります。

- 「用語の確認」
- 「セキュリティ」
- 「旧バージョンのデータベースと DDF のバージョン」
- 「DDF Builder におけるエラーの検出と修正」
- 「元の定義と変更済み定義の保存」
- 「テーブル定義エディター ページ」
- 「DDF Builder におけるレガシー ヌル」
- 「DDF Builder における真のヌル」

用語の確認

DDF Builder で作業を開始する前に、このマニュアルで使用している用語についてもう一度確認してください。用語には、リレーショナル データベース モデルでのみ使用されるものもあれば、トランザクショナル データベース モデルに関するものもあります。DDF Builder はトランザクショナル データからリレーショナル テーブル定義を作成するよう設計されているため、これら 2 つのモデルを一つの場所で一緒に用います。トランザクショナル モデルで使用される用語がリレーショナル モデルでは異なる意味になることもあるため、ここでの情報は役に立ちます。次の表に、このマニュアルで使用される用語を示します。

用語	定義
データベース	DDF のセット、テーブル定義、データ ファイルのセットおよび DSN を含めたファイルの集合体。
ファイル テーブル	Btrieve 物理ファイルまたは SQL テーブル名。
辞書 データ辞書 システム カタログ システム テーブル	データ（メタデータ）のテーブル定義を含めることができる DDF ファイルのセット。
テーブル定義	Btrieve ファイルに対応する DDF のエントリ。
DDF システム テーブル システム オブジェクト	拡張子が .ddf のファイル。DDF は、リレーショナル データベース（メタデータ）の制約内でファイルを定義するための手段を提供するシステム テーブルです。
キー インデックス	特定の順序でテーブルまたはファイルをソートしたり、特定の値の検索を最適化するために指定されるデータ。

セキュリティ

DDF Builder でセキュリティが有効なデーターベースを扱う場合は、DDF Builder でそのファイルを開く前にすべてのセキュリティを無効にしておくことをお勧めします。セキュリティを無効にできなかった場合、DDF Builder では呼び出しが実行されるたびに安全なログインとパスワードを要求します。呼び出しごとにこの要求があると、テーブル定義の作成や変更に必要な回数によっては作業効率の低下を招く恐れもあります。

旧バージョンのデータベースと DDF のバージョン

DDF Builder は PSQL v9 以上のバージョンでサポートされます。Scalable SQL v4.00 以上で作成された DDF は Btrieve データ ファイル v6.x 以上と一緒にサポートされます。

Scalable SQL v3.xx 以前のバージョン

Scalable SQL v3.xx 以前のバージョンで作成された DDF がある場合は、それを DDF Builder で開く前に変換しておく必要があります。

この手順については、「レッスン 1 ― v3.00 の DDF を使った作業」を参照してください。

Btrieve v5.x 以前のバージョン

DDF Builder ではバージョン 6.x のファイルをサポートしますが、バージョン 6.x より前に作成されたファイルはサポートしません。Btrieve v5.x 以前に作成された Btrieve データ ファイルがある場合は、それを DDF Builder で開く前にリビルドしておく必要があります。

この手順については、「レッスン 2 ― v6.x より前のファイル形式での作業」を参照してください。

DDF Builder におけるエラーの検出と修正

Btrieve ファイルに関連付けられている既存のテーブル定義を開く場合、DDF Builder では一連の比較とチェックを行います。これらの比較とチェックによってエラーが検出されると、DDF Builde はテーブル定義に対して一定の調整を行うことがあります。テーブル定義に対する変更は定義エラー ビューに記録されます。

メモ DDF Builder では必ず元のテーブル定義を保持するので、DDF Builder で行った変更がテーブル定義に自動的に保存されてしまうことはありません。変更されたテーブル定義または元のテーブル定義のどちらかを別名で保存すれば両方の定義を保持できます。

DDF Builder が行う変更の評価と可能性を理解できるよう、DDF Builder がたどる手順を見てみましょう。

まず、DDF Builder は Btrieve ファイルを調べて、レコード長、インデックス、インデックス セグメントなど適切な内容を分析します。次に、DDF Builder は既存の DDF を開き、その DDF に含まれる情報を見て Btrieve ファイルと比較します。双方の内容が互いに一致するかしないか、体系的に比較されます。

DDF Builder では、Btrieve ファイルで見つかった内容を基準にしてエラーや問題を検出および修正します。次の例でわかりやすく説明します。

例 1 ― インデックスの不一致

1 つのキーを含む Btrieve ファイルがあります。テーブル定義にはこれに対応するインデックスがありません。DDF Builder では Btrieve ファイルを変更することはできないので、この Btrieve ファイルで定義されているキーがインデックスとしてテーブル定義へ引き継がれます。

この種の状況を処理する場合の情報については、「レッスン 6 – インデックスの不一致」を参照してください。

例 2 ― レコード長の不一致

Btrieve ファイルには総レコード長が 120 バイトで定義されているレコードがありますが、テーブル定義では 100 バイト分しか定義されていませんでした。DDF Builder では Btrieve レコードを変更できないので、テーブル定義の方に、割り当てられていない 20 バイトの列を追加します。

この種の状況を処理する場合の情報については、「レッスン 8 – レコード長の不一致」を参照してください。

例 3 ― フラグの不一致

Btrieve ファイルのキーには、テーブル定義内の対応する SQL インデックスのフラグ設定と一致しないフラグがあります。前の例と同様、DDF Builder では Btrieve ファイルを変更できないので、テーブル定義の SQL インデックスにおけるフラグ設定は Btrieve ファイルのフラグ設定と一致するよう変更されます。

この種の状況を処理する場合の情報については、「レッスン 5 ― ファイル / フィールド フラグの不一致」を参照してください。

メモ 「DDF Builder チュートリアル」の章では、DDF Builder で作業時に発生する可能性があるさまざまな状況に対応する手順を提供します。

DDF Builder ではすべての問題を修正することはできません。DDF Builder で検出され修正される問題については、「テーブル定義エラー」を参照してください。

元の定義と変更済み定義の保存

DDF Builder は既存のテーブル定義に対して推奨する変更を行いますが、その変更を保持するよう要求されたり、あるいは元のテーブル定義を破棄するよう要求されることはありません。

変更した定義または元の定義のどちらかを別名で保存すれば、両方の定義を保存しておくことができます。DDF Builder によって行われた変更を受け入れる場合、変更したテーブル定義か元のテーブル定義のどちらかを別名で保存できます。

DDF Builder によって行われた変更を拒否する場合、元のテーブル定義は元の名前で保持されます。

テーブル定義エディター ページ

テーブル定義エディターは、テーブル定義のビュー、作成および変更に使用する複数の情報ページで構成されます。テーブル定義エディターには以下のページがあります。

- ［テーブル］ページ
- ［インデックス］ページ
- ［プレビュー］ページ
- ［統計情報］ページ
- ［SQL ビュー］ページ

［テーブル］ページ

［テーブル］ページはテーブル定義の作成や変更時に大半の作業を行うページです。この［テーブル］ページには、**未加工データ ビュー**と**グリッド データ ビュー**があります。

未加工データ ビュー

未加工データ ビューでは、Btrieve ファイルのデータを ASCII と 16 進数表示で見ることができます。このビューにはレコード長、オフセット、およびフィールド サイズも表示されます。

また、未加工データ ビューでは列、ヌル インジケーターの判断や、不明なフィールドやバイトを特定するために必要なビジュアル インジケーターを提供します。

ヒント 未加工データ ビューで使用されるビジュアル インジケーターの詳細については、「未加工データ ビューでのフィールド属性」を参照してください。

グリッド データ ビュー

グリッド データ ビューの機能は、Zen Control Center の SQL Editor で見られるグリッド ウィンドウ ビューに似ています。

インデックスに使用されるフィールドはサイズや型を変更することができません。フィールド名のみグリッド データ ビューで変更できます。インデックスの追加、インデックスの削除、またはその他のインデックス情報の変更を行うには、Btrieve Maintenance ユーティリティを使用して Btrieve ファイルを直接変更する必要があります。

ヒント グリッド データ ビューで使用されるビジュアル インジケーターの詳細については、「グリッド データ ビューでのフィールド属性」を参照してください。

[インデックス] ページ

[インデックス] ページでは、DDF Builder が Btrieve ファイルで検出したインデックスとインデックス セグメントを見ることができます。このページからインデックスやインデックス セグメントを追加したり変更したりすることはできません。[インデックス] ページで可能なのはインデックス名の変更のみです。[テーブル] ページでインデックス フィールドの名前を変更した場合、その変更は [インデックス] ページでリアルタイムに反映されインデックス名が更新されます。

メモ インデックスの名前付けに必要な手順については、「インデックスの名前を付ける」を参照してください。

オルタネート コレーティング シーケンス（ACS）ファイル

オルタネート コレーティング シーケンス（ACS）ファイルを使用する Btrieve ファイルで作業する場合、その ACS ファイルは作業対象の Btrieve ファイルと同じディレクトリにあり、拡張子は .alt（upper.alt など）である必要があります。

[プレビュー] ページ

[プレビュー] ページでは、現在のテーブル定義を使用して書式設定されたファイルのデータを見ることができます。テーブル定義を変更すると、それに応じてこのプレビューも変わります。

[プレビュー] ページは読み取り専用です。このページに表示される情報を編集することはできません。ただし、このページの下部にある矢印ボタンを使ってレコードを移動することはできます。

[統計情報] ページ

[統計情報] ページでは、テーブル定義エディター で開いたファイルの Btrieve ファイル統計情報が表示されます。このページに表示される情報は、Btrieve Maintenance ユーティリティでこのファイルに対して生成された統計情報レポートで報告される情報と同じです。

[SQL ビュー] ページ

[SQL ビュー] ページでは、テーブル定義の作成や変更を反映した SQL ステートメントが表示されます。テーブル定義を変更すると、その基となる SQL ステートメントは [SQL ビュー] ページで直ちに更新されます。

このページの情報は読み取り専用なので、このページ上で内容の変更を行うことはできません。必要であれば、ステートメントを選択してコピーし別の場所で使用してください。

注意　Zen 辞書システム オブジェクトへの参照が含まれるステートメントは再利用しないでください。これらのオブジェクトは X$< **テーブル名** > という書式で簡単に特定され、所有者から再利用を禁止するコメントが記述されています。

DDF Builder におけるレガシー ヌル

レガシー ヌルについては、Pervasive.SQL 2000 より前の Zen のバージョンでのみサポートしていました。レガシー ヌルを使用するファイルは DDF Builder で認識されますが、DDF Builder で作業する場合は、このタイプのヌルを処理するために必要なアクションはありません。

DDF Builder における真のヌル

Pervasive.SQL 2000 では真のヌルに対するサポートが追加されました。真のヌルについては、フィールドの先頭にあるバイトで、ヌル インジケーター バイトとしても知られています。これは、該当する列にヌル値の可否を指定するために用いられます。

真のヌルを使った作業

DDF Builder でレコード フィールドを定義する場合、レコード内にヌル値を許可するフィールドがあるかどうかを知っていることが重要です。なぜなら、レコードの一部分でヌル値を許可することにした場合、1 バイトが余分にフィールドへ追加されるからです。ファイルは Btrieve ファイルであるため、レコードでヌル値が許可される部分は、ヌル インジケーター バイトを使用することで指定されます。ヌル インジケーター バイトは、テーブル定義エディター ページの未加工データ ビューでフィールドまたは列の直前のバイトに該当します。[ヌル]チェック ボックスが選択されると、ヌル インジケーター バイトがアクティブになり、そのヌル インジケーター バイト分を確保するためフィールドのサイズは自動的に 1 バイト削減されます。

メモ　Pervasive.SQL 2000 より前に作成されたファイルで作業する場合、ヌルはほとんど問題になりません。

ヌル値を許可するフィールドの作成

DDF Builder でヌルを許可するフィールドを作成する際は、フィールドのサイズとヌル インジケーター用に必要な 1 バイトを考慮する必要があります。これはヌル値が可能なフィールドのサイズを 25 バイトにするつもりであれば、実際は 26 バイトとして定義し、1 バイト分をヌル インジケーター用に確保することを意味します。フィールドに対してヌル値を許可するよう指定すると、そのフィールドのサイズは自動的に DDF Builder によって 1 バイト削減されます。

メモ　ヌル値を許可する列とヌル値を許可しない列での作業例については、チュートリアル 1 の例で「未加工データ ビューでヌル値を許可する列を作成する」と「グリッド データ ビューでヌル値不可の列をヌル値を許可する列に変更する」を参照してください。

GUI のリファレンス

DDF Builder ではグラフィカル ユーザー インターフェイス（GUI）が利用できます。この GUI には、オブジェクトの表示や作業に使用するさまざまなエディター、ビューおよびウィザードがあります。

編集中のオブジェクトは、エディター上部にタブによって示されます。タブにはオブジェクトの名前が表示されます。エディター内で変更したデータは明示的に保存する必要があります。

ビューは、一度に 1 つのみ開くことができます。ビュー内で実行された操作は直ちに適用されます。明示的に保存する必要はありません。

ウィザードでは 1 つまたは複数のダイアログによって特定の結果を得るための作業手順を導きます。

以下の表に DDF Builder で提供されるエディター、ビューおよびウィザードを示します。

表 3　DDF Builder のエディター、ビューおよびウィザード

GUI コンポーネント	エディター	ビュー	ウィザード	説明
データ ソース エクスプローラー		○		「データ ソース エクスプローラー」
Btrieve ファイル エディター	○			「Btrieve ファイル エディター」
テーブル定義エディター	○			「テーブル定義エディター」
SQL Editor	○			『*Zen User's Guide*』の 「SQL Editor」[1]
グリッド ウィンドウ ビュー		○		『*Zen User's Guide*』の「グリッド ウィンドウ ビュー」[1]
テキスト ウィンドウ ビュー		○		『*Zen User's Guide*』の「テキスト ウィンドウ ビュー」[1]
アウトライン		○		『*Zen User's Guide*』の「アウトライン ウィンドウ ビュー」[1]
データベースの追加			○	「データベースの追加」
テーブルの一貫性のチェック（データベースの確認およびテーブルの確認）		○ 2		「テーブルの一貫性のチェック」
SQL 定義のコピー			○	「SQL 定義のコピー」
Btrieve スキーマのエクスポート			○	「Btrieve スキーマのエクスポート」
Btrieve スキーマのインポート			○	「Btrieve スキーマのインポート」
関連するデータファイルの変更			○	「関連するデータ ファイルの変更」
Btrieve 型		○		「Btrieve 型」
定義エラー		○		「定義エラー」
元の定義		○		「元の定義」

[1] DDF Builder と ZenCC では同じコンポーネントを共有しています。このため、共有するエディター、ビューおよびウィザードについてはほかの Zen マニュアルで説明しており、この『*DDF Builder User's Guide*』では説明していません。

[2] データベース チェックやテーブルの一貫性チェック アクションはデータ ソース エクスプローラーで選択を行うことを除けば、ダイアログではなくウィザードのようなものです。一貫性のチェックによって生成される結果はビューに似ています。

［ようこそ］ページ

DDF Builder を起動すると［ようこそ］タブが表示されます。このタブでは、さまざまな情報へアクセスしたり、新しいサーバーの追加や新しいデータベースの作成などの共通タスクを実行したりすることができます。提供される情報は、以下のとおりです。

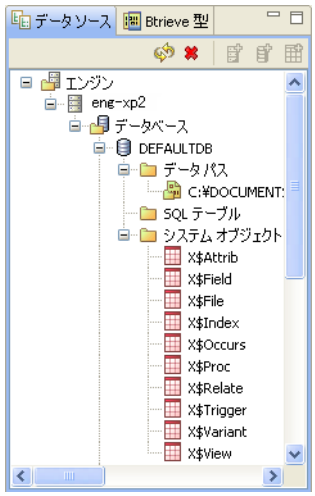
- Getting Started with Zen DDF Builder
- Zen DDF Builder チュートリアル
- FAQ
- 共通タスク

［ようこそ］タブが表示されていない場合は、DDF Builder メニューから［**ヘルプ**］>［**ようこそ**］を選択して表示させることができます。

データ ソース エクスプローラー

DDF Builder では**データ ソース エクスプローラー**と呼ばれるオブジェクト ツリー形式のファイル エクスプローラーを使用します。オブジェクトには、データベース、データ パス、Btrieve ファイルおよび SQL テーブルなどがあります。これらのオブジェクトはツリー内で**ノード**と呼ばれることもあります。

図 2 データ ソース エクスプローラー



ノードは展開したり折りたたんだりして、下位ノードの表示 / 非表示を切り替えることができます。ノードに下位ノードがある場合は、そのノードの左側に展開 / 折りたたみ用アイコンが表示されます。

Btrieve ファイル エディター

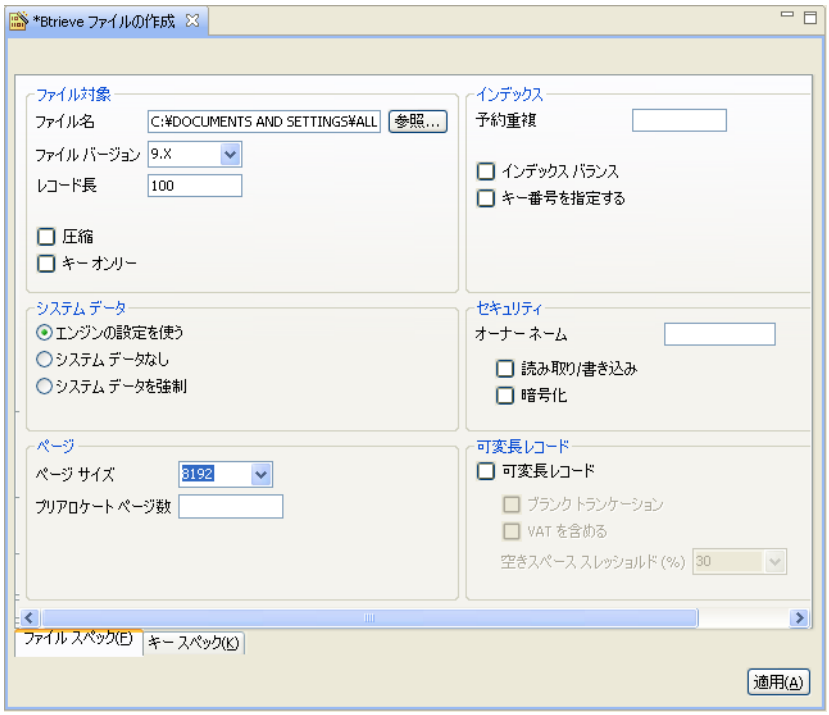
Btrieve ファイル エディターでは新しい Btrieve ファイル用のファイル仕様およびキー仕様を作成します。DDF は新しい Btrieve ファイルに対して自動的に作成されません。DDF が必要な場合は別途追加する必要があります（「SQL テーブル」を参照）。

ヒント Btrieve ファイルおよびその DDF を作成するには、Control Center の Zen を使用します。

アクセス方法

データ ソース エクスプローラーで、［**データ パス**］ノードの直下にある作業対象のデータ パス ノードを右クリックし、［**Btrieve ファイルの作成**］を選択します。

図 3 Btrieve ファイル エディター



Btrieve ファイルは物理ストレージに置く必要があります。これは Btrieve ファイル エディターがデータ パスから起動されるからです。

機能

Btrieve ファイル エディターには、ファイルの特性を指定するタブとキーの特性を指定するタブがあります。各タブにある［**適用**］ボタンをクリックすると、そのタブで指定した仕様が保存されます。［**ファイル**］>［**保管**］をクリックすると両方のタブで行った変更を一括保存します。

Btrieve ファイルを作成するためには、少なくともファイル名とレコード長を指定する必要があります。これは［ファイル スペック］タブで指定します。

「必要な知識」でも述べているように、このマニュアルは、トランザクショナル アクセス方法とリレーショナルの概念について十分に理解していることを前提として説明しています。このセクションでは Btrieve ファイル エディターのタブにあるコントロールについては説明しません。Btrieve ファイルについてさらに理解を深める必要がある場合は、次の表で提供する関連ドキュメントを参照してください。

表 4　Btrieve ファイルの作成に関連するドキュメント

タブ	上級者リファレンス マニュアル	開発者リファレンス マニュアル
ファイル スペック	『*Advanced Operations Guide*』 「ファイル サイズ」 「オーナー ネームおよびセキュリティ」 「システム データ」 「ファイル情報エディター」	『*Zen Programmer's Guide*』 「MicroKernel エンジンの基礎」の章 「データベースの設計」の章
キー スペック	『*Advanced Operations Guide*』 「重複キーの操作方法」	『*Zen Programmer's Guide*』 「MicroKernel エンジンの基礎」の章 「データベースの設計」の章 「レコードの処理」の章 「データベースの作成」の章

制限事項

一度に作成できる Btrieve ファイルは 1 つだけです。[キー スペック] タブで、セグメント キーのセグメントを追加または変更したら［**適用**］をクリックします。これにより、各セグメントへの変更が保存され、次のセグメントを新たに作成または編集することができます。

テーブル定義エディター

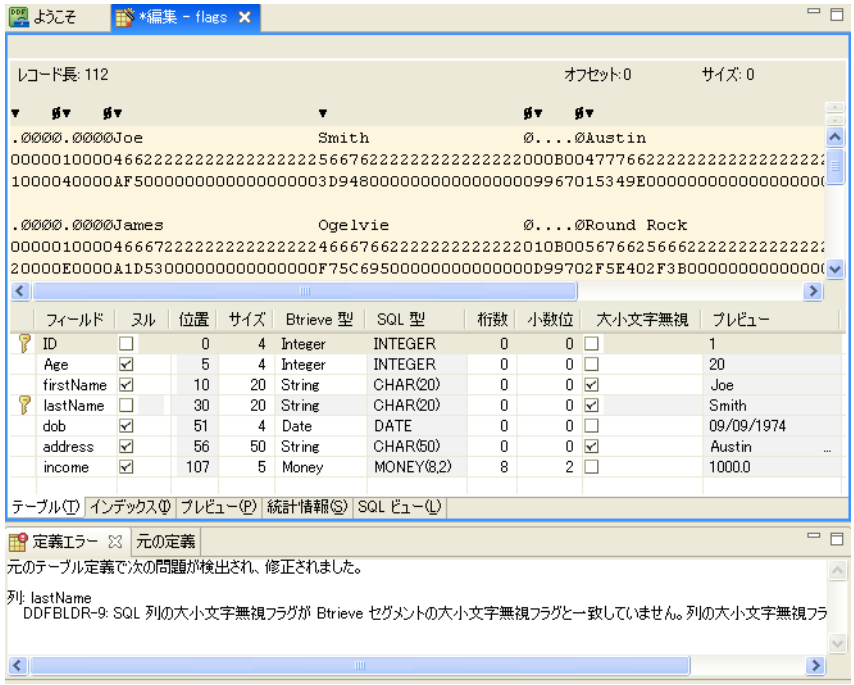
テーブル定義エディターでは、SQL メタデータ用のスキーマを新規作成したり既存のスキーマを変更したりします。テーブル定義エディターは、Zen Control Center（ZenCC）の Table Editor と似ています。詳細については、「Table Editor」を参照してください。

アクセス方法

データ ソース エクスプローラーで以下の操作のいずれかを実行します。

- Btrieve ファイル名を右クリックし、［**テーブル定義の作成**］を選択します。
- SQL テーブル名をダブルクリックし、既存のテーブル定義を編集します。
- SQL テーブル名を右クリックし、［**テーブル定義の編集**］を選択します。

図 4 テーブル定義エディター



機能

テーブル定義エディターは、Zen Control Center（ZenCC）の Table Editor と同様に、タブまたはページを使用してテーブル定義情報のさまざまなビューを表示します。読み取り専用の情報ページもあれば、テーブル定義を作成したり編集したりする作業領域を持つページもあります。

制限事項

列が Btrieve ファイルのキーとして使用されている場合、その列のサイズ、オフセットおよびデータ型を変更することはできません。DDF Builder では既存の Btrieve ファイルのレイアウト構造を変更できません。

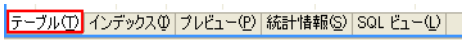
［テーブル］ページ

テーブル定義エディターの［テーブル］ページでは、2 種類のメタデータ ビュー（未加工ビューとグリッド ビュー）を提供します。未加工ビューでは情報を Btrieve ファイルで表示し、グリッド ビューは情報を SQL テーブルとして表示します。

アクセス方法

［テーブル］ページはテーブル定義エディターの下部にある［**テーブル**］タブをクリックして開きます。

図 5 テーブル定義エディターの［テーブル］ページ タブ

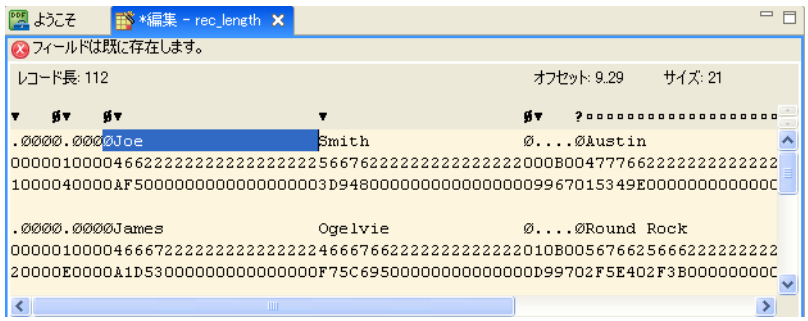


未加工データ ビュー

未加工データ ビューではデータ レコードの 16 進値と ASCII 値を同時に表示します。また、このビューではレコード長のほか、オフセットや選択したバイト数も表示します。このオフセットやサイズの値は、ビュー内で選択したバイト位置やバイト数に応じて調整されます。

未加工データ ビューの上部にはフィールド、列、ヌルおよび不明であることを示すインジケーターが表示されます。フィールド インジケーターは、レコード内で各フィールドの開始位置を示します。ヌル インジケーターはヌル インジケーター バイトが置かれる場所を示します。不明なフィールドやバイトも未加工データ ビューで表示されます。

図 6 テーブル定義エディターの未加工データ ビュー



次に未加工データ ビューで表示される属性を挙げます。

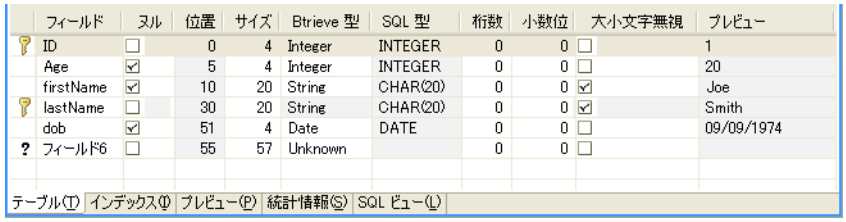
表 5　未加工データ ビューでのフィールド属性

属性	説明
エラー メッセージ	エラーや警告の状態を表示します。
レコード長	レコード全体の長さを表示します。可変長部分は除外されます。
位置（オフセット）	選択したバイトまたはフィールドの先頭および末尾の位置を表示します。
サイズ	選択したバイトまたはフィールドのサイズ（バイト単位）を表示します。
データ インジケーター	フィールド/列インジケーター（▼）は、レコード内における各フィールドの開始位置を示します。 フィールドがヌル値可能として指定されている場合は、ヌル インジケーター（§）を表示します。このバイトはヌル インジケーター バイトであることを表します。 フィールドが未定義の場合は、不明なフィールド インジケーター（?）を表示します。 バイトが未定義の場合は、不明なバイト インジケーター（□）を表示します。

グリッド データ ビュー

グリッド データ ビューは、テーブルのスキーマ構造を行と列のグリッドとして表示します。各フィールドはグリッド上で行として表されます。各行は、各フィールドの属性を示すセルで構成されます。属性の多くは編集可能で、スキーマへの変更として保存もできます。

図 7 テーブル定義エディターのグリッド データ ビュー



次にグリッド データ ビューで表示される属性を挙げます。

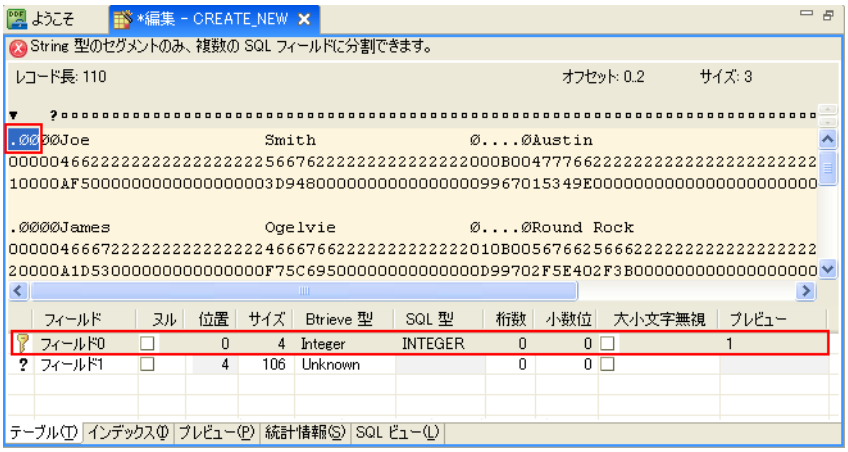
表 6　グリッド データ ビューでのフィールド属性

属性	説明
列インジケーター	フィールドがキー（インデックス）定義に使用される場合は、キー アイコン（ ）が表示されます。 フィールドが不明な場合は、不明であることを示すアイコン（ ）が表示されます。 フィールドが不明な可変長部分である場合は、可変アイコン（ ）が表示されます。
フィールド	テーブルのフィールド名を表示します。
位置（オフセット）	オフセット、つまり、レコード内のフィールドの位置を基準点 0 からの差で表した値を表示します。 表示のみ
サイズ	フィールドのサイズ（バイト単位）を表示します。
Btrieve 型	フィールドの Btrieve データ型を表示します。
SQL 型	フィールドの SQL データ型を表示します。 表示のみ
桁数	浮動小数点数値の場合、有効な桁数を表示します。
小数位	浮動小数点数値の場合、小数点の右側に現れる有効な桁数を表示します。
ヌル	フィールドでヌル値を使用する場合は選択します。
大小文字無視	フィールドで大文字と小文字を区別しない場合は選択します。
プレビュー	適用したデータ型で書式設定されたフィールドの内容を表示します。 表示のみ

2 つのビューで一緒に作業する方法

グリッド データ ビューでセルを選択すると、未加工データ ビューではそのセルのフィールドに該当するバイト部分が示されます。同様に、未加工データ ビューで任意のバイト部分を選択すると、グリッド データ ビューではそれに該当するフィールド行が選択されます。

図 8 テーブル定義エディターでのバイトの選択



1 つのフィールドの定義範囲を越えて次のフィールド定義まで選択しようとすると、エラー メッセージが表示され、グリッド データ ビューではその両方の行が選択されます。エラー メッセージと警告メッセージは共に未加工データ ビューで表示されます。

図 9 テーブル定義エディター エラー メッセージ

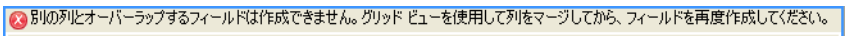
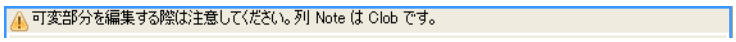


図 10 テーブル定義エディターの注意 / 警告メッセージ



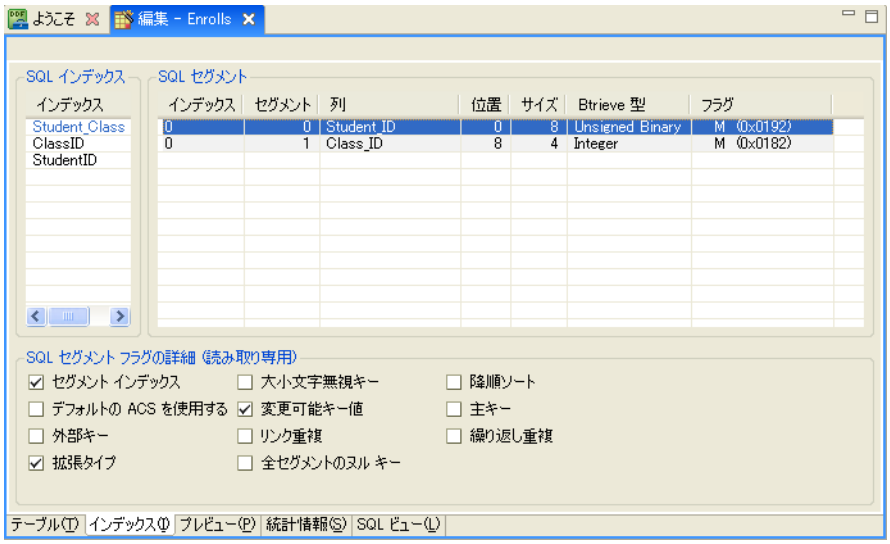
[インデックス] ページ

[インデックス] ページの内容は、SQL インデックス名を除き読み取り専用です。このページで、インデックスの構造を変更することはできません。インデックスの追加や SQL テーブルへの変更は Zen Control Center で行う必要があります。『*Zen User's Guide*』の 「Table Editor」を参照してください。

アクセス方法

[インデックス] ページはテーブル定義エディターの下部にある [**インデックス**] タブをクリックして開きます。

図 11 テーブル定義エディターの［インデックス］ページ



機能

［テーブル］ページで、インデックスとして指定されているフィールド名を変更すると、その変更はすぐに［インデックス］ページにも反映されます。

制限事項

［インデックス］ページでは SQL インデックス名のみが変更できます。

オルタネート コレーティング シーケンス（ACS）ファイルを使用する Btrieve ファイルで作業する場合、その ACS ファイルは作業対象の Btrieve ファイルと同じディレクトリにあり、拡張子は .alt（upper.alt など）である必要があります。

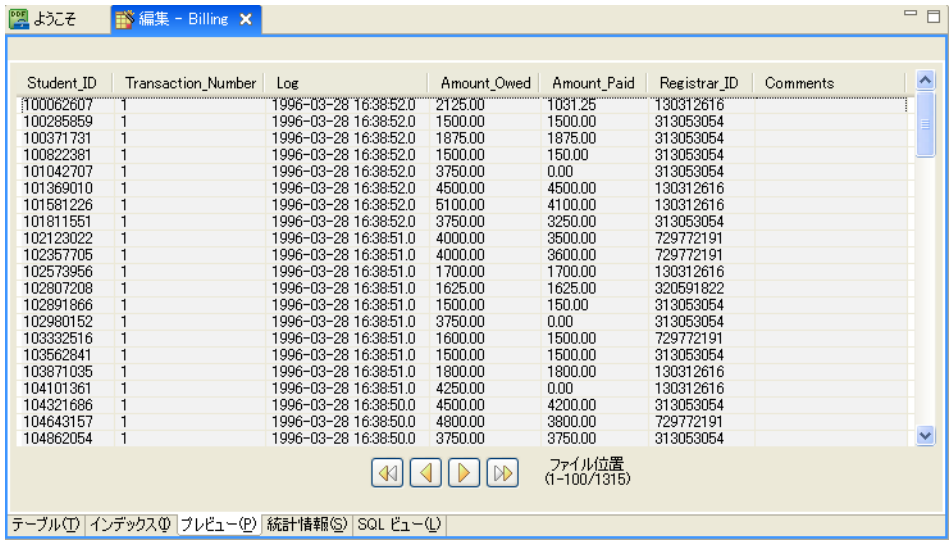
［プレビュー］ページ

［プレビュー］ページは、ファイルのデータを行と列で構成した読みやすいレイアウトで表示します。

アクセス方法

［プレビュー］ページはテーブル定義エディターの下部にある［**プレビュー**］タブをクリックして開きます。

図 12 テーブル定義エディターの［プレビュー］ページ



機能

ページの下部中央にあるボタンを使用するとデータ レコード間を移動できます。このボタンの右側にはファイルの位置が表示されます。ファイル位置は、総レコード数のうち表示されているレコード範囲を示します。たとえば、"100-199/1314" と表示されている場合、総レコード数 1,314 件のうち、100 番目から 199 番目までのレコードが表示されていることを示します。

［テーブル］ページで列定義を変更すると、その変更はすぐに［プレビュー］ページにも反映されます。

このページの情報は読み取り専用で、変更できません。

［統計情報］ページ

［統計情報］ページには Btrieve ファイルのファイル仕様とキーの仕様が表示されます。この情報は読み取り専用で、このビュー内では変更できません。

アクセス方法

［統計情報］ページはテーブル定義エディターの下部にある［**統計情報**］タブをクリックして開きます。

図 13 テーブル定義エディターの［統計情報］ページ

機能

統計情報は、Btrieve ファイルの構造上の特性を見るための便利な手段です。これは、ファイルのスキーマをエクスポートしようと考えているがファイルやキーの仕様がよくわからないときには大いに役立ちます。

「Btrieve スキーマのエクスポート」も参照してください。

メモ 統計情報は、単に物理的な Btrieve ファイルの情報を基にしています。メタデータの情報を示すわけではありません。

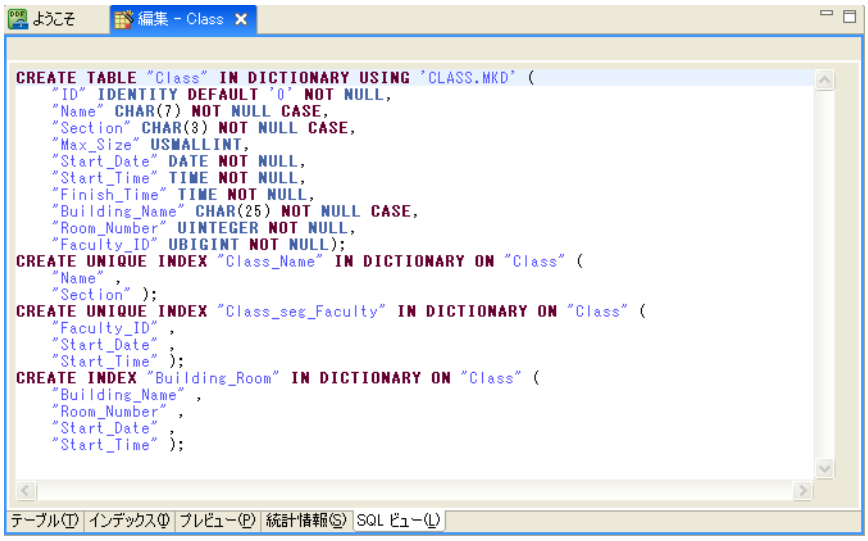
［SQL ビュー］ページ

［SQL ビュー］ページでは、現在のテーブル定義を作成するために必要な SQL ステートメントを表示します。

アクセス方法

［SQL ビュー］ページはテーブル定義エディターの下部にある［**SQL ビュー**］タブをクリックして開きます。

図 14 テーブル定義エディターの［SQL ビュー］ページ



機能

DDF Builder で作成される SQL ステートメントに基づいて新しいスキーマを作成するために、ここで作成されている大部分の SQL ステートメントをコピーして Zen Control Center で再利用することができます。Zen Control Center で使用する SQL ステートメントは将来使用するため保存しておくこともできます。

制限事項

ここで作成される SQL ステートメントはコピーして再利用することができますが、保存はされません。また、現在 DDF Builder で提供されている SQL ステートメントを自動的に保存する手段もありません。SQL ステートメントを保存するには、このウィンドウからそのステートメントをコピーしてテキスト エディターに貼り付けて保存する必要があります。

注意 Zen 辞書システム オブジェクトへの参照が含まれるステートメントは再利用しないでください。これらのオブジェクトは X$< **テーブル名** > という書式で簡単に特定され、所有者から再利用を禁止するコメントが記述されています。

データベースの追加

これはデータベースを新規作成します。このダイアログは Zen Control Center（ZenCC）と共有になっています。『*Zen User's Guide*』の「データベース」および「［データベースの新規作成］GUI のリファレンス」を参照してください。

テーブルの一貫性のチェック

DDF Builder ではテーブルの一貫性をチェックする機能を提供します。一貫性のチェックでは一連の検証ルールを使用して、物理データ ファイルとメタデータをそのデータ辞書ファイルで比較します。

一貫性のチェックでは以下のような条件を検証します。

- 辞書ファイルには、互換性のあるバージョンの情報があること。
- テーブルに、有効な名前、ID および既存のアクセス可能な物理ファイルがあること。
- 列は、有効な名前があり、合計レコード長が正しく、定義が重複していないこと。
- インデックスは、名前、数、サイズ、データ型、オフセットおよび属性において正しいこと。

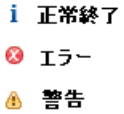
一貫性のチェックは、一度にすべてのテーブルを対象に行ったり、テーブルを個別に選択して行うこともできます。このチェックでは、オブジェクトによる検証メッセージ、エラーおよび警告のカウントを報告します。オブジェクトとはデータベース、テーブルまたはデータ辞書ファイル（DDF）です。

" 正常終了 " という見出しによって分類される検証メッセージは、検証された一貫性チェックを一覧表示します。エラーは " エラー " という見出しによって分類され、警告は " 警告 " という見出しによって分類されます。

エラー メッセージがある場合は必ず表示されます。オブジェクトにエラーと警告の両方がある場合、警告は " エラー " にもリストされます。

検証メッセージや警告は表示したり非表示にしたりすることができます。

結果ビューのアイコンによってメッセージのさまざまなタイプが識別されます。



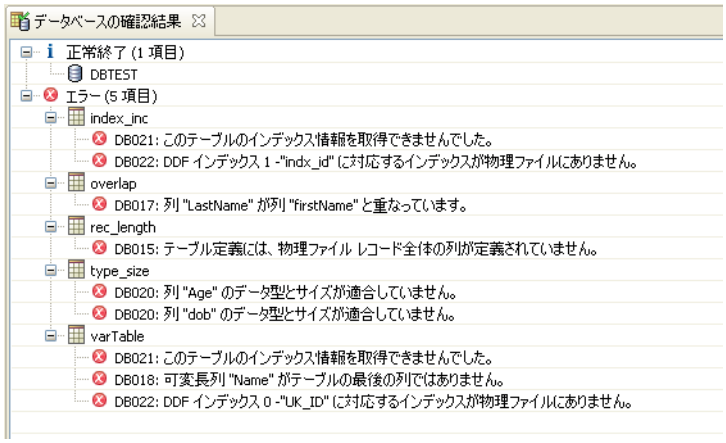
エラーはテーブル定義における問題を示すもので、多くの場合、データ ファイルがアクセスされたときにエラーや不正なデータが返される原因になります。たとえば、DDF に定義されているインデックスがデータ ファイルにはないとエラーになります。そのようなインデックスは実際にはデータ ファイルにないため、エンジンがその特定のインデックスで最適化するような SQL クエリではエラーが発生します。

警告は起こり得る問題を暗示するものですが、その問題がエラーを引き起こすとは限りません。たとえば、あるインデックスがデータ ファイルに定義されているが、対応する DDF エントリがない場合は警告になります。SQL アクセスはインデックスについて知らないので、それを使用しようとはしません。そのため時間のかかるクエリになるかもしれませんが、このクエリは最終的には正しい結果を返します。

アクセス方法

データ ソース エクスプローラーで、データベース名を右クリックして［**データベースの確認**］を選択するか、［**データ パス**］または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックして［**テーブルの確認**］を選択します Shift キーまたは Ctrl キーを押したまま対象のテーブル名をクリックすると、複数のテーブルを選択することができます。

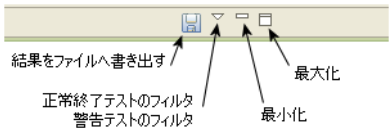
図 15 DBCheck（一貫性チェックの結果）ビュー



機能

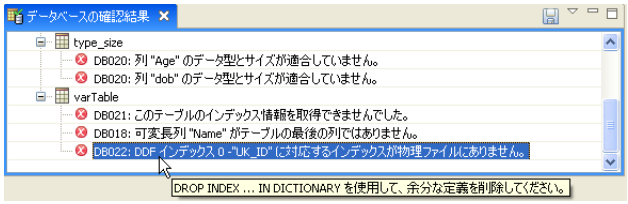
DBCheck ペインはドッキング解除して DDF Builder ウィンドウ内の別の場所へ移動させることができます。このペインは最小化および最大化することができます。

また、このペインでは一貫性チェックの結果をテキスト ファイルに保存したり、検証チェックのメッセージや警告をフィルター処理で非表示にするアイコンも提供します。



エラー メッセージのヒント

エラー メッセージ上でクリックするとヒントが表示されます。このヒントではメッセージに関するコメントを提供し、場合によっては修正アクションが提案されることもあります。DBCheck の結果をファイルに保存する場合は、このヒントもそのファイルに書き込まれます。



SQL 定義のコピー

SQL 定義のコピー ウィザードでは、既存の SQL テーブルのスキーマをソースとして使用し、新たな SQL テーブルを作成します。また、このウィザードでは新しい SQL テーブルと関連付けられる Btrieve ファイルも作成されます。

アクセス方法

データ ソース エクスプローラーで SQL ファイル名を右クリックし、[**SQL 定義のコピー**] を選択します。

機能

このウィザードでは以下の項目について指定が必要です。

- データベース エンジンが起動しているサーバー名
- 新規テーブルを作成するデータベース
- 新規テーブルの名前。デフォルトでは、関連付けられる Btrieve ファイルの名前になります。既存のテーブル名は使用できません。

メモ 新しいテーブルにはデータがありませんが、ソース テーブルからデータをエクスポートして、新しいテーブルにインポートすることはできます。「データ インポート ウィザードを使ったデータのインポート」および「データ エクスポート ウィザードを使ったデータのエクスポート」を参照してください。

Btrieve スキーマのエクスポート

Btrieve スキーマのエクスポート ウィザードでは、ソースとなる Btrieve ファイルのスキーマの仕様を定める XML ファイルを作成します。この XML ファイルを使用することによって、既存の Btrieve ファイルの構造に基づいた Btrieve ファイルを新たに作成することができます。「Btrieve スキーマのインポート」を参照してください。

ソース ファイルのデータはエクスポートされません。データをエクスポートする場合は、「データ エクスポート ウィザードを使ったデータのエクスポート」を参照してください。

アクセス方法

データ ソース エクスプローラーで、[**データ パス**] ノードの下位にある Btrieve ファイル名を右クリックし、[**Btrieve スキーマのエクスポート**] をクリックします。

機能

デフォルトで、このウィザードではエクスポートされる XML ファイル名にソース ファイル名を使用します。この XML 出力ファイルは .xml というファイル拡張子が付けられ、ソース ファイルと同じディレクトリに書き込まれます。出力ファイルのデフォルト名前と保存場所は変更することもできます。

また、エクスポートする前に XML の内容をプレビューすることもできます。

Btrieve スキーマのインポート

Btrieve スキーマのインポート ウィザードでは、別の Btrieve ファイルの構造を基にして Btrieve ファイルを作成します。この構造は XML フォーマットによるソース ファイルのスキーマである必要があります。

ソース ファイルのデータはインポートされません。データをインポートする場合は、「データ インポート ウィザードを使ったデータのインポート」を参照してください。

アクセス方法

データ ソース エクスプローラーで、[**データ パス**] ノードの直下にある作業対象のデータ パス ノードを右クリックし、[**Btrieve スキーマのインポート**] を選択します。

Btrieve ファイルは物理ストレージに置く必要があります。これはこのウィザードがデータ パスから起動されるからです。

機能

このウィザードは、既知のデータ パスのコンテキストから起動するため、ターゲット ファイルのパスを指定する必要はありません。ファイル名を指定するだけです。ファイル名の長さは 255 文字以下で指定してください。[スペースを含むファイル / ディレクトリ名] クライアント設定オプションが有効になっていない場合、名前にスペースを含めることはできません。デフォルトでこのオプションは有効になっています。『*Advanced Operations Guide*』で、クライアント用の「アプリケーション特性」設定にある「スペースを含むファイル / ディレクトリ」プロパティも参照してください。

データ パスの追加

データ パスは、データ ソース エクスプローラーで [**データ パスの追加**] コマンドを使用して追加します。データ パスとは、Btrieve ファイルが存在する物理ストレージ上の場所を指します。各データベースには、識別されるデータ パスが最低 1 つあります。

Btrieve ファイル用に作成される DDF は、そのデータベースの元のデータ パスに置かれます。これはそのデータベース全体の SQL テーブルがすべて同じ DDF のセットに定義されるためです。

アクセス方法

データ ソース エクスプローラーで、[**データ パス**] ノードを右クリックし、[**データ パスの追加**] をクリックします。

機能

既存のディレクトリは、空の場合と既にファイルが含まれている場合があります。

[削除] コマンドを使用すると、データ ソース エクスプローラーからデータ パスが削除されます（削除対象の**データ パス**またはデータベース名を右クリックし、[**削除**] をクリックします）。ディレクトリは物理ストレージから削除**されません**。

関連するデータ ファイルの変更

DDF Builder では、選択したテーブル定義（SQL テーブル）に関連付けられるデータ ファイルを変更することができます。

アクセス方法

データ ソース エクスプローラーで、関連付けされているデータ ファイルを変更する SQL テーブルを右クリックし、［**関連するデータファイルの変更**］をクリックします。関連付けるデータ ファイルの完全なパス名を入力するか、参照して選択します。

機能

特定の SQL テーブルに関連付けられているデータ ファイルを変更する場合、ファイル名を入力するか、［参照］ボタンを使用してファイルの場所を参照し選択します。

Btrieve 型

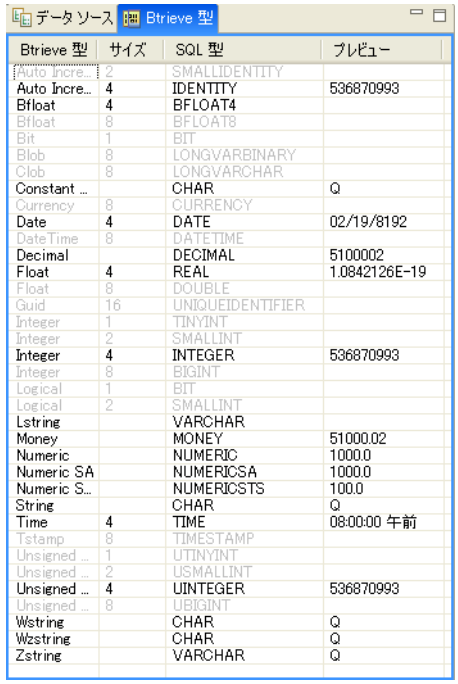
DDF Builder では、Btrieve データ型とサイズ、およびそれに対応する SQL データ型を表示する単独のビュー（Btrieve ビュー）を提供しています。

この Btrieve 型ビューはデータの分析に使用することができます。未加工データ ビューで任意のバイトを選択すると、Btrieve 型ビューではそのバイト（サイズ）に適合するデータ型でデータがどのように見えるかを表示します。テーブル定義エディターで特定の列を選択した場合にも、Btrieve 型ビューでは書式設定されたデータのプレビューを表示します。

アクセス方法

［Btrieve 型］タブは DDF Builder ウィンドウの左側のペインのデータ ソース エクスプローラーで［データ ソース］タブの隣にあります。［**Btrieve 型**］タブをクリックすると、Btrieve 型ビューが表示されます。

図 16 Btrieve 型ビュー



Btrieve 型	サイズ	SQL 型	プレビュー
Auto Incre...	2	SMALLIDENTITY	
Auto Incre...	4	IDENTITY	536870993
Bfloat	4	BFLOAT4	
Bfloat	8	BFLOAT8	
Bit	1	BIT	
Blob	8	LONGVARBINARY	
Clob	8	LONGVARCHAR	
Constant ...		CHAR	Q
Currency	8	CURRENCY	
Date	4	DATE	02/19/8192
DateTime	8	DATETIME	
Decimal		DECIMAL	5100002
Float	4	REAL	1.0842126E-19
Float	8	DOUBLE	
Guid	16	UNIQUEIDENTIFIER	
Integer	1	TINYINT	
Integer	2	SMALLINT	
Integer	4	INTEGER	536870993
Integer	8	BIGINT	
Logical	1	BIT	
Logical	2	SMALLINT	
Lstring		VARCHAR	
Money		MONEY	51000.02
Numeric		NUMERIC	1000.0
Numeric SA		NUMERICSA	1000.0
Numeric S...		NUMERICSTS	100.0
String		CHAR	Q
Time	4	TIME	08:00:00 午前
Tstamp	8	TIMESTAMP	
Unsigned ...	1	UTINYINT	
Unsigned ...	2	USMALLINT	
Unsigned ...	4	UINTEGER	536870993
Unsigned ...	8	UBIGINT	
Wstring		CHAR	Q
Wzstring		CHAR	Q
Zstring		VARCHAR	Q

機能

Btrieve 型ビューのペインは、データを比較したりプレビューしたりして不正なデータ型やサイズを解決しようとする際に役立ちます。このタスクをサポートするために、Btrieve 型ビューのペインをドッキング解除して DDF Builder ウィンドウ内の適当な位置に移動させることができます。

［プレビュー］列も提供され、選択したデータがその特定のデータ型に対してどのように表示されるかを示します。また、この列ごとのプレビューはデータ型のサイズを決定する場合にも利用できます。

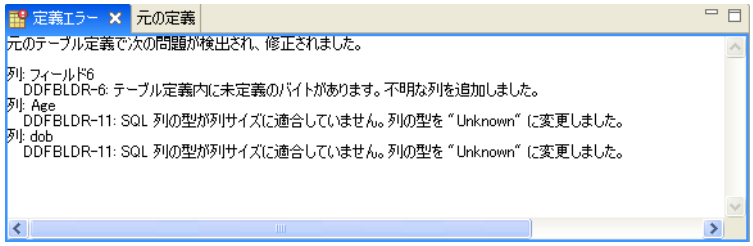
定義エラー

DDF Builder では、既存のテーブル定義で問題を検出して変更するときは自動的に定義エラー ビューが開きます。このウィンドウを閉じた場合、再度このウィンドウを開けば変更を見直すことができます。

アクセス方法

DDF Builder のメニュー バーで［**ウィンドウ**］>［**定義エラーの表示**］を選択します。エラーがなければ、このコマンドは淡色表示になります。

図 17 テーブル定義エディターの定義エラー



機能

定義エラー ビューに挙げられているエラーは DDF Builder で解決される問題に関する具体的な情報を示しています。元のテーブル定義も読み取り専用モードで元の定義ペインに提供されます。これらのペインは比較しやすいように並べて開くことができます。

定義エラー リスト

次の表では、DDF Builder が既存の定義で検出する可能性のあるテーブル定義エラーを示します。発生したエラーの一般的な説明と、それを受けて DDF Builder がどのようにテーブル定義を修正するかについても示します。

表 7 テーブル定義エラー

表示されるメッセージ	意味
DDFBLDR-1：この SQL インデックスに対応する Btrieve キーがありません。インデックスを無視しました。	既存のテーブル定義の SQL インデックスに対応するキーが Btrieve ファイルにありません。 SQL テーブルのインデックスは無視されます。
DDFBLDR-2：この Btrieve キーに対応する SQL インデックスが見つかりません。インデックスを追加しました。	既存のテーブル定義の SQL インデックスとして定義されていないキーが Btrieve ファイルにあります。 Btrieve ファイルにあるキーに対応するインデックスをテーブル定義に追加します。
DDFBLDR-3：SQL セグメントのヌル フラグが、対応する Btrieve セグメントのヌル フラグと一致していません。	SQL ファイル セグメントにないヌル フラグを持つセグメントが Btrieve ファイルに含まれています。 または Btrieve ファイルにないヌル フラグを持つセグメントが SQL ファイルに含まれています。
DDFBLDR-4：SQL 列がほかの列にオーバーラップしています。列のサイズを切り詰めました。	テーブル内のほかの列と重複している SQL 列が既存のテーブル定義にあります。
DDFBLDR-5：Btrieve セグメントに対応する SQL 列が見つかりません。新しい列を追加しました。	既存のテーブル定義の SQL 列に対応しないセグメントが Btrieve ファイルにあります。
DDFBLDR-6：テーブル定義内に未定義のバイトがあります。不明な列を追加しました。	Btrieve ファイルで見つかったバイト数と同じバイト数が既存のテーブル定義で割り当てられていません。このテーブル定義には定義されていないバイトがあります。 不明な列が新たにテーブル定義へ追加され、それが未定義のバイト部分に割り当てられます。
DDFBLDR-7：SQL 列の型が Btrieve セグメントの型と一致していません。列の型を変更しました。	Btrieve ファイル セグメントが SQL テーブルの列で検出されたデータ型とは異なるデータ型を使用しています。 SQL テーブルの列のデータ型は Btrieve ファイルのセグメントと同じデータ型に変更されます。

表 7　テーブル定義エラー

表示されるメッセージ	意味
DDFBLDR-8：SQL 列のヌル フラグが Btrieve セグメントのヌル フラグと一致していません。列のヌル フラグを変更しました。	DDF Builder が Btrieve セグメントで検出したものとは異なるヌル フラグ設定を持つ SQL 列が既存のテーブル定義にあります。
DDFBLDR-9：SQL 列の大小文字無視フラグが Btrieve セグメントの大小文字無視フラグと一致していません。列の大小文字無視フラグを変更しました。	DDF Builder が Btrieve セグメントで検出したものとは異なる大小文字無視フラグ設定を持つ SQL 列が既存のテーブル定義にあります。
DDFBLDR-10：SQL 列が Btrieve セグメントの境界を越えて定義されています。列を無視しました。	Btrieve ファイルのセグメントの位置およびサイズと合致しない SQL 列が既存のテーブル定義にあります。 セグメントの位置およびサイズと合致しない SQL 列は既存のテーブル定義で無視されます。
DDFBLDR-11：SQL 列の型が列サイズに適合していません。列の型を "Unknown" に変更しました。	既存のテーブル定義に、列のサイズと適合しないデータ型の SQL 列が含まれています。 SQL 列のデータ型は不明に変更されます。サイズは変わりません。
DDFBLDR-12：可変長列にできるのはテーブル内の最後の列だけです。列の型を "Unknown" に変更しました。	既存のテーブル定義では、テーブルの最後ではない列に可変長部分を定義しています。 SQL 列のデータ型は不明に変更されます。
DDFBLDR-13：不正なビット マスクが検出されました。適切なビット マスクに変更しました。	バイトを割り当てるために必要なサイズと異なる不正なビット マスクがテーブル定義に含まれています。ビット マスクはテーブル定義で変更されるので、そのサイズは修正されバイトを足し合わせます。
DDFBLDR-14：Btrieve WSTRING および WZSTRING と互換性のある SQL 型はありません。列の型を SQL CHAR にセットしました。列は使用できません。	WSTRING 型または WZSTRING 型で定義されている列が Btrieve ファイルに含まれています。これらのデータ型は SQL データ型に割り当てられません。これらのデータ型の列は使用できないため CHAR 型に変更されます。

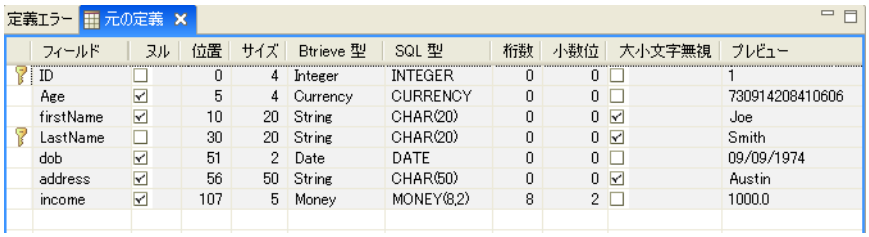
元の定義

DDF Builder が問題を検出して既存のテーブル定義を変更する場合、変更を保存するまでは元のテーブル定義が保持されます。ユーザーが既存のテーブル定義に変更を加える場合も、その変更を保存するまでは元の定義が保持されます。元のテーブル定義は、元の定義ビューを開いて見ることができます。

アクセス方法

DDF Builder のメニュー バーで［**ウィンドウ**］>［**元の定義の表示**］を選択します。変更がなければ、このメニュー コマンドは淡色表示になります。

図 18 テーブル定義エディターの元の定義



	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
🔑	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
	Age	☑	5	4	Currency	CURRENCY	0	0	☐	730914208410606
	firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
🔑	LastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
	dob	☑	51	2	Date	DATE	0	0	☐	09/09/1974
	address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
	income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

機能

元のテーブル定義は読み取り専用モードでこのビューに提供されます。このビューは、更新されたテーブル定義と比較しやすいよう移動することができます。

変更した定義または元の定義のどちらかを別名で保存すれば、元のテーブル定義をずっと保持することができます。これを行うには、［**ファイル**］>［**別名保管**］をクリックして別の名前を入力します。

DDF Builder の各種作業

このセクションでは、DDF Builder で実行する作業について説明します。作業は以下のカテゴリに分かれています。

カテゴリ	説明
「一般的な作業」	DDF Builder の使用法全般の説明です。
「データ ソース エクスプローラーから開始する作業」	エディター、ビューおよびウィザードにアクセスできます。

一般的な作業

一般的な作業は、このツールの全体的な使用に適用されます。

▶▶ DDF Builder を起動するには

「DDF Builder の起動」を参照

ユーザー マニュアルへのアクセス

▶▶ DDF Builder のオンライン ヘルプにアクセスするには

1 DDF Builder を起動します。

2 関心があるユーザー インターフェイス上でクリックします。

3 F1 を押します。

4 関心があるユーザー インターフェイス以外の別の領域上をクリックした場合は、再度 F1 を押してヘルプ ビューを更新する必要があるかもしれません。

［ヘルプ］>［Zen ドキュメント ライブラリ］を選択することもできます。DDF Builder と Zen Control Center（ZenCC）は同じコンポーネント（SQL Editor など）を共有しているので、DDF Builder 内で開くオンライン ヘルプであっても Zen の他の機能に関する情報が表示されることもあります。

DDF Builder ログ ファイル

▶▶ DDF Builder ログ ファイルにアクセスするには

1 DDF Builder を起動し、［ヘルプ］>［DDF Builder のログを表示］をクリックします。

このログ ファイルは、システムで使われているテキスト エディターで開きます。

▶▶ DDF Builder ログ ファイルをクリアするには

1 DDF Builder を起動し、［ヘルプ］>［DDF Builder のログの消去］をクリックします。

この操作を確認するダイアログ ボックスが開きます。

2 削除の確認で［OK］をクリックするとログ ファイルが削除されます。

次回 DDF Builder を起動したときには、新しいログ ファイルが自動的に作成されます。

▶▶ データ ソース エクスプローラーでノードが追加または削除されたことを確認するには

1 追加または削除を確認するノードを右クリックします。

2 ［更新］をクリックします。

［更新］コマンドのコンテキストは、このコマンドを呼び出したノードに適用されます。データ ソース エクスプローラー内にあるすべてのノードを更新する場合は、［エンジン］ノード（ツリーの最上位）からこのコマンドを実行します。

データ ソース エクスプローラーから開始する作業

DDF Builder における作業の多くは、データ ソース エクスプローラー内にあるノードに関連付けられているコマンドを選択することによって実行します。コマンドによってエディター、ビューまたはウィザードが起動します。次に示す表では、希望するアクションを実行するための手順を説明します。この表は、関連する分野ごとにまとめられています。

- 「Btrieve ファイル」
- 「データ」
- 「データベース」
- 「データ ソース エクスプローラー」
- 「オンライン ヘルプ」
- 「SQL クエリ」
- 「SQL テーブル」

メモ DDF Builder でセキュリティが有効なデーターベースを扱う場合は、DDF Builder でそのファイル作業を行う前に必ずデータベースをオフラインにし、すべてのセキュリティを無効にしておきます。

Btrieve ファイル

実行したいアクション	必要な作業	注記
エディターで Btrieve ファイルを作成する	［**データ パス**］ノードの下にあるデータ パス名を右クリックします。 ［**Btrieve ファイルの作成**］をクリックします。	DDF Builder は、新規作成した Btrieve ファイルの DDF を自動的に作成しません。 「Btrieve ファイル エディター」を参照してください。
Btrieve ファイルの構造を XML ファイルへエクスポートする	**データ パス名**の下にある Btrieve ファイル名を右クリックします。 ［**Btrieve スキーマのエクスポート**］をクリックします。	Btrieve ファイルのデータ自体はエクスポート**されません**。 「Btrieve スキーマのエクスポート」を参照してください。
別の Btrieve ファイルの XML で指定されている構造を基にして新しいファイルを作成する	［**データ パス**］ノードの下にあるデータ パス名を右クリックします。 ［**Btrieve スキーマのインポート**］をクリックします。	Btrieve ファイルのデータ自体はインポート**されません**。 「Btrieve スキーマのインポート」を参照してください。
Btrieve ファイルを置くディレクトリを指定する	［**データ パス**］フォルダー を右クリックします。 ［**データ パスの追加**］をクリックします。	Btrieve ファイル用に作成される DDF は、そのデータベースの元のデータ パスに置かれます。これはすべての SQL テーブルが同じ DDF のセットに定義されるためです。 「データ パスの追加」を参照してください。
データ ソース エクスプローラーからデータ パス ディレクトリを削除する	［**データ パス**］ノードの下にあるデータ パス名を右クリックします。 ［**削除**］をクリックします。	データ パスにあるディレクトリやファイルは物理ストレージから削除されません。

実行したいアクション	必要な作業	注記
SQL テーブルが関連付けられている Btrieve ファイルの統計情報を表示する	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［**テーブル定義の編集**］をクリックします。 ［**統計情報**］タブをクリックします。	テーブル名は、［**データ パス**］ノードの下、または［**SQL テーブル**］ノードの下に表示されます。 「［統計情報］ページ」を参照してください。
選択した SQL テーブルに関連付けられている Btrieve（データ）ファイルを変更する	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［**関連するデータ ファイルの変更**］をクリックします。	特定のテーブルに関連付けられるデータ ファイルは 1 つだけです。

データ

実行したいアクション	必要な作業	注記
すべてのテーブルの一貫性を一度にチェックする	**データベース** ノードを右クリックします。 ［**データベースの確認**］をクリックします。	「テーブルの一貫性のチェック」を参照してください。
1 つまたは複数のテーブルの一貫性を個別にチェックする	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［**テーブルの確認**］をクリックします。	テーブル名は、［**SQL テーブル**］ノードの下、または Btrieve ファイルに DDF がある場合は Btrieve ファイル名のノードの下に表示されます。 （Shift キーまたは Ctrl キーを押したまま対象のテーブル名をクリックすると、複数のテーブルを選択することができます）。 「テーブルの一貫性のチェック」を参照してください。
データ ファイルを置くディレクトリ、またはデータ ファイルが既に存在するディレクトリをデータ ソース エクスプローラーで指定する	［**データ パス**］フォルダー を右クリックします。 ［**データ パスの追加**］をクリックします。	「データ パスの追加」を参照してください。
データ ソース エクスプローラーからデータ ファイルが存在するディレクトリを削除する	［**データ パス**］ノードの下にあるデータ パス名を右クリックします。 ［**削除**］をクリックします。	データ パスにあるディレクトリやファイルは物理ストレージから削除されません。

データベース

実行したいアクション	必要な作業	注記
データベースの新規作成	**データベース** ノードを右クリックします。 ［**データベースの追加**］をクリックします。	「データベースの追加」を参照してください。
データ ソース エクスプローラーからデータベースを削除する	**データベース** ノードを右クリックします。 ［**削除**］をクリックします。	データベースは物理ストレージからは削除されません。

データ ソース エクスプローラー

実行したいアクション	必要な作業	注記
ツリーを更新してノードへの追加、またはノードから削除を確認する	SQL テーブル名を除く任意のノードを右クリックします。 ［**更新**］をクリックするか、F5 キーを押します。	［更新］コマンドのコンテキストは、このコマンドを呼び出したノードに適用されます。データ ソース エクスプローラー内にある**すべての**ノードを更新する場合は、ツリーの最上位にある［**エンジン**］ノードからこのコマンドを実行します。

オンライン ヘルプ

実行したいアクション	必要な作業	注記
ユーザー向けドキュメントへのアクセス	エディター、ビューまたはウィザード内で F1 キー（Linux の場合は Shift + F1 キー）を押すか、［**ヘルプ**］>［**PSQL ドキュメント ライブラリ**］をクリックします。	「ユーザー マニュアルへのアクセス」の説明を参照してください。

SQL クエリ

実行したいアクション	必要な作業	注記
SQL テーブルのすべてのレコードに対して SELECT ステートメント（SELECT * FROM）を実行する または SQL テーブルに対して実行する SQL ステートメントを入力できるエディターにアクセスする	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［SQL Editor］をクリックします。	デフォルトで、SQL Editor は開いたときに SELECT * FROM ステートメントを実行します。 SQL テーブル名は［**SQL テーブル**］ノード、または Btrieve ファイル名（Btrieve ファイルに DDF がある場合）のノードの下に表示されます。 「データベースの追加」を参照してください。
SQL Editor またはアウトライン ビューで SQL ステートメントを実行する	『*Zen User's Guide*』の「実行のタスク」を参照してください。	SQL Editor では多数のタスクを実行できます。詳しい説明については、『*Zen User's Guide*』の「SQL Editor の各種作業」を参照してください。

SQL テーブル

実行したいアクション	必要な作業	注記
エディターでテーブルを新規作成する	**データ　パス名**の下にある Btrieve ファイル名を右クリックします。 ［**テーブル定義の作成**］をクリックします。	Btrieve ファイル用に 1 つまたは複数の SQL テーブルを作成できます。各 SQL テーブルは Btrieve ファイル名のノードの下に表示されます。 「テーブル定義エディター」を参照してください。
別の SQL テーブルのスキーマを基にしてテーブルを新規作成する	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［**SQL 定義のコピー**］をクリックします。	テーブル名は、［**SQL テーブル**］ノードの下、または Btrieve ファイルに DDF がある場合は Btrieve ファイル名のノードの下に表示されます。 「SQL 定義のコピー」を参照してください。
テーブル定義を変更する	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［**テーブル定義の編集**］をクリックします。	テーブル名は、［**SQL テーブル**］ノードの下、または Btrieve ファイルに DDF がある場合は Btrieve ファイル名のノードの下に表示されます。 「テーブル定義エディター」を参照してください。
データ　ソース　エクスプローラーからテーブルを削除する	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［**削除**］をクリックします。	テーブルは物理ストレージからは削除されません。 テーブル名は、［**SQL テーブル**］ノードの下、または Btrieve ファイルに DDF がある場合は Btrieve ファイル名のノードの下に表示されます。
SQL テーブルに関連付けられている Btrieve（データ）ファイルを作成する	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［**関連するデータ ファイルの変更**］をクリックします。	特定のテーブルに関連付けられるデータ ファイルは 1 つだけです。
SQL テーブルが関連付けられている Btrieve ファイルの統計情報を表示する	Btrieve ファイル名の下、または［**SQL テーブル**］ノードの下にある SQL テーブル名を右クリックします。 ［**テーブル定義の編集**］をクリックします。 ［**統計情報**］タブをクリックします。	テーブル名は、［**データ パス**］ノードの下、または［**SQL テーブル**］ノードの下に表示されます。 「［統計情報］ページ」を参照してください。

DDF Builder チュートリアル

3

DDF Builder を使用するためのサンプル ガイド

以下のトピックでは DDF Builder を使用するための学習資料を提供します。

DDF Builder チュートリアルの使用

この章では、DDF Builder を使用した 2 つのチュートリアルを提供します。各チュートリアルでは、データの構造に関する基本的な情報、ファイルを使用したサンプル シナリオ、チュートリアルの目的、および各演習の目的を達成するために必要となる一般的な手順を提供します。

メモ DDF Builder でテーブル定義を作成または変更するには、データの構造についてある程度の知識が必要です。Btrieve ファイルの列の定義、またフィールドのオフセットやサイズを知っていると役に立ちます。DDF Builder では列定義の構造を導こうと試みますが、どの程度それを提供できるかについては限りがあります。

チュートリアル 1 の概要

最初のチュートリアルでは、Btrieve ファイルはあるが、リレーショナル アクセスを可能にするために必要なテーブル定義がないという状況を解決します。Btrieve ファイル用の DDF がないため、まずは DDF を作成しなければいけません。このチュートリアルでは、空の DDF を使用してデータベースを新規作成し、Btrieve ファイル用のテーブル定義をその DDF に追加する手順を示します。

また、このチュートリアルの実行を通して、DDF Builder インターフェイスで作業するための基本的な知識も提供します。このチュートリアルで初めて DDF Builder を使用する場合は、第 2 章「DDF Builder の使用」をもう一度読んで、インターフェイスやコンポーネントが置かれている場所についてよく理解しておくとよいでしょう。

チュートリアル 2 の概要

2 番目のチュートリアルでは、既存のテーブル定義に古い定義あるいは不正な定義情報が含まれているため、そのテーブル定義を修正するという少し複雑な状況を解決します。このチュートリアルでは、既存のテーブル定義を見直して Btrieve ファイルのセットに関連付けられている DDF に対して必要な変更を行う手順を示します。

この 2 番目のチュートリアルはさまざまなレッスンで構成されています。各レッスンでは DDF Builder を使用して DDF を変更しようとする場合に発生する可能性がある問題に焦点を当てています。DDF Builder で自動的に解決される問題もありますが、手動で変更を行う必要がある問題もあります。

事前の確認

チュートリアルを開始する前に行うことがいくつかあります。このセクションでは、本章のチュートリアルを開始する前に行っておく必要がある事項を確認します。

ファイルのバックアップ

DDF Builder でテーブル定義を作成したり変更することによって、データベースの構造が変更されます。用心のために、作業対象のファイルを必ずバックアップしておいてください。この作業対象ファイルにはデータ ファイルと既存の辞書ファイルの両方が含まれます。

DDF Builder では Btrieve ファイルを開きますが、DDF への変更を行うだけで既存の Btrieve ファイルへの変更は行いません。しかし、すべてのファイルをバックアップしておくことや、コピーしたファイルで作業することをお勧めします。DDF の部分的な変更によって既存のテーブル定義が破損してしまい、作業中の定義が壊れて使用できなくなることもあります。

メモ リレーショナル アクセスを提供する既存の Btrieve ファイル**および** DDF を変更するつもりであれば、DDF Builder を使用する必要はありません。Zen Control Center を使用すれば、DDF や Btrieve ファイルを変更することができます。

DDF Builder では IN DICTIONARY 呼び出しを使用してテーブル定義を DDF へ書き込みます。DDF Builder を使用した場合は、Btrieve ファイルに書き込みが行われることはありません。

データ ファイルをバックアップするか、作業対象ファイルのコピーを作成したら、チュートリアルで使用されるファイルの所在を確認する準備が整います。

チュートリアル ファイルの所在確認

DDF Builder のインストール時には、この章に含まれているチュートリアルを使用するためのフォルダーやファイルがシステム上に作成されます。このチュートリアル ファイルはデフォルトの Application Data ディレクトリで以下の場所にインストールされます。

<*Application Data*>¥DDFBuilder	DDF Builder アプリケーション ファイル
<*Application Data*>¥DDFBuilder¥tutorials¥tutorial1	チュートリアル 1 のファイル
<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2¥v3	チュートリアル 2 のファイル（レッスン 1 とレッスン 2）
<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2	チュートリアル 2 のファイル（レッスン 3 から 8）

メモ Zen のデフォルトのインストール ディレクトリの詳細については、『*Getting Started with Zen*』の「ファイルはどこにインストールされますか？」を参照してください。

データ ソース名（DSN）の作成

DDF Builder 内でチュートリアルのデータベースにアクセスするには、各データベースに対して関連付けられたデータ ソース名（DSN）が作成されていること確かめておく必要があります。

チュートリアル 1 では、データベースを作成する時点で DSN を作成します。チュートリアル 1 は Btrieve ファイルしかない状態から始めるため、データベースも、DDF、DSN もすべて 1 から作成する必要があります。

チュートルアル 2 では、DDF Builder インターフェイス内でファイルとフォルダーを見るために 2 つの DSN を作成する必要があります。一方の DSN はチュートリアル 2 のデータベースを指し、もう一方の DSN は Tutorial2 フォルダーにある V3 データベースを指します。

次の表では ODBC データ ソース アドミニストレーターで Tutorial 2 および V3 データベースの DSN を作成するためのオプションを示します。

データ ソース名	説明	データベース名
Tutorial2	ODBC アクセス	Tutorial2
V3	ODBC アクセス	V3

ヒント ODBC データ ソース アドミニストレーターを使って DSN を作成する必要がある場合は、『*Zen User's Guide*』を参考にしてください。

ファイルをバックアップし、チュートリアルのファイルの所在を確認し、DSN を作成したら、チュートリアル 1 を開始する準備が整いました。

チュートリアル 1 ー DDF Builder でテーブル定義を作成する

シナリオ

このチュートリアルでは、Btrieve データ ファイルがあり、そのレポートを作成するために ODBC アクセスを提供する必要があります。

これを行うには、その Btrieve ファイル用のテーブル定義を作成する必要があります。テーブル定義はデータベースの DDF ファイルに格納されます。ご存知のとおり、この Btrieve ファイル用のテーブル定義はないので最初から作成します。

目的

このチュートリアルの目的は、テーブル定義のない Btrieve ファイルへリレーショナル アクセスを提供するために必要な手順を実践することです。このチュートリアルの目的を達成するためには、DDF Builder を使用して以下の作業を実行します。

- ❒ Zen データベースを作成する
- ❒ 既存の Btrieve ファイルを開く
- ❒ DDF Builder による調査結果を検証する
- ❒ レコード フィールドを定義する
- ❒ テーブル定義を保存する
- ❒ インデックス情報を検証する
- ❒ 定義したデータをプレビューする

ヒント テーブル定義の作成手順を通して、DDF Builder を使用するための基本的な作業を学習します。このチュートリアルは、手順の中でその基本的な作業を実践するよう設計されているので、DDF Builder を使用するための一般的な作業についてよく理解することができます。

必要な知識

DDF Builder でテーブル定義の作成や変更作業を行うときは、そのデータの構造についてある程度の知識が必要です。このレコード長やインデックス情報などいくつかの情報は Btrieve ファイル自身に保存されます。しかし、列情報は Btrieve ファイルに保存されません。この列情報はソフトウェア ベンダーやアプリケーション開発者から提供される場合もあります。データを入念に調べてテーブル定義の作成や変更を試みることもできますが、データの構造について十分な知識を持っていない場合はこのユーティリティを使用しないようにしてください。

このチュートリアルの場合、データの構造はわかっています。このファイルの構造を見てみましょう。

CREATE_NEW.MKD という名前の Btrieve データ ファイルがあります。このファイルのレコード長は 110 バイトで 6 個のフィールドがあります。このフィールドは次のように定義されています。

表 8　作成する DDF のデータ構造

フィールド	サイズ	データ型	桁数	小数位	ヌル	大小文字無視	インデックス
ID	4	Integer			×		○
First Name	20	String			○	○	
Last Name	20	String			×	○	
DOB	4	Date			○		
Address	50	String			○	○	
Income	8	Currency	8	2	○		

このファイルには、ID フィールドに設定された重複のないキー（一意のインデックス）があり、そのインデックスには indx_id という名前が付けられています。

データ構造について理解できたら、作業を開始することができます。DDF Builder を使用して空のデータベースを作成することから始めます。これは Btrieve ファイル用に定義する空の DDF のセットを提供します。

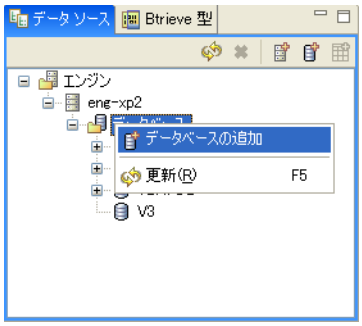
Zen データベースを作成する

既存の Btrieve ファイル用に DDF を作成する前に、まずはデータベースを作成する必要があります。DDF Builder をまだ実行していない場合は起動します。「DDF Builder の起動」を参照してください。

▶▶ データベースを作成するには

1　データ　ソース　エクスプローラーで、データベースを作成するマシンのツリーを展開します。これは　DDF Builder がインストールされている同じマシンです。

2　［**データベース**］アイコンを右クリックし、次に［**データベースの追加**］をクリックします。



新しいデータベースを作成するための［データベースの新規作成］ダイアログ ボックスが開きます。

3　［データベースの新規作成］ウィザードで、以下のパラメーターを指定します。

- データベース名：Tutorial1
- 場所：

  <*Application Data*>¥DDFBuilder¥tutorials¥tutorial1¥
- バウンド：チェックしない
- 辞書ファイルの作成：チェックする
- 関係整合性の設定：チェックする

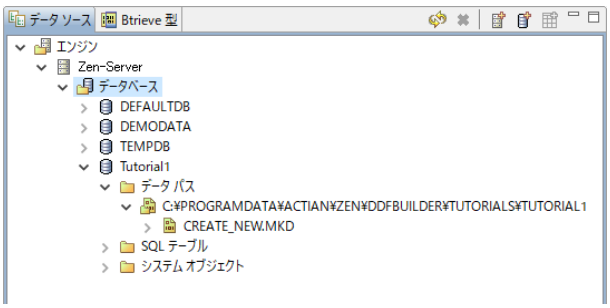
- 長いメタデータ（V2 メタデータ）：チェックする
- データベース コード ページ：サーバーのデフォルト
- 32 ビット エンジン DSN の作成：チェックする

［データベースの新規作成］ダイアログは次のようになります。

ヒント　これらの手順でデータベースを作成すると、データ ソース名（DSN）も作成されます。チュートリアルの作業が終了したら、ここで作成された DSN とデータベースは削除してもかまいません。

4　［**完了**］をクリックすると、指定した場所にデータベースと空の DDF ファイルが作成されます。

データベースが作成されると、データ ソース エクスプローラーのデータベース ツリーにそのデータベースが新しいノードとして現れます。



このデータベース ノードには、［データ パス］、［SQL テーブル］および［システム オブジェクト］フォルダーがあります。［データ パス］フォルダーには Btrieve ファイルの場所が含まれています。［SQL テーブル］フォルダーにはデータベースのリレーショナル テーブルが含まれ（この時点ではありません）、［システム オブジェクト］フォルダーはシステム辞書テーブル用および、この場合は空のデータ辞書ファイル用のフォルダーになります。

これらの要素を使用して Zen データベースを作成することが、Btrieve ファイル用のテーブル定義を作成する最初の手順です。先程作成されたデータベースには辞書ファイル（DDF）がありますが、現時点でこれらのファイルには DDF 自身の構造用の定義しか含まれていません。DDF Builder で Btrieve ファイルを開いたら、このデータベース用に作成された DDF へテーブル定義を追加し、その Btrieve ファイルのデータにアクセスできるようにします。

このデータベースには SQL テーブルが含まれていないので注意してください。含まれているのはこのチュートリアルで使用する CREATE_NEW.MKD Btrieve ファイルのみです。これは定義済みのユーザー テーブルがないからです。データ ファイルとは別に、空の DDF が［システム オブジェクト］フォルダーに置かれています。テーブル定義を作成すると、そのテーブル定義に一致する SQL テーブルも作成されます。この SQL テーブルはテーブル定義にアクセスします。引き続き、DDF Builder で Btrieve ファイルを開く作業に進みます。

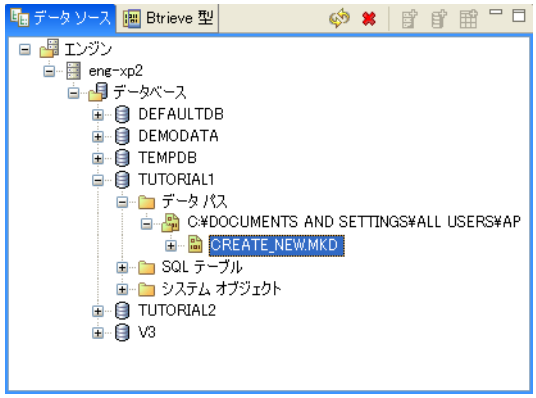
既存の Btrieve ファイルを開く

Btrieve ファイルと同じ場所に DDF ファイルを作成したら、DDF Builder でそのファイルを開いてデータ構造がどのように解読されるかを見ます。

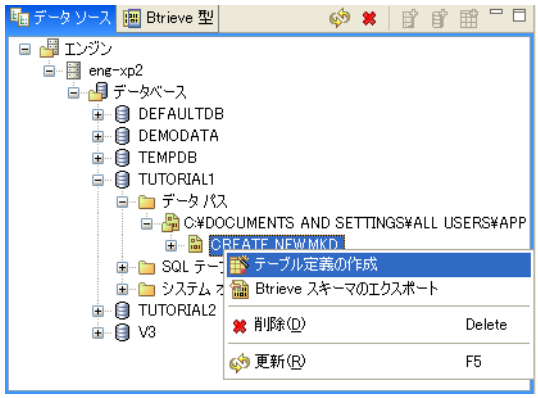
▶▶ Btrieve ファイルを開くには

1　DDF Builder で、ローカル マシンを選択し、作成したデータベースを選択します。

データ ソース エクスプローラーはデータ パスの一覧を表示して、mkd ファイルがインストールされたディレクトリを示します。これはデータベースを作成した場所と同じディレクトリです。データ ソース エクスプローラーで TUTORIAL1 データベース ディレクトリの構造は次のようになります。



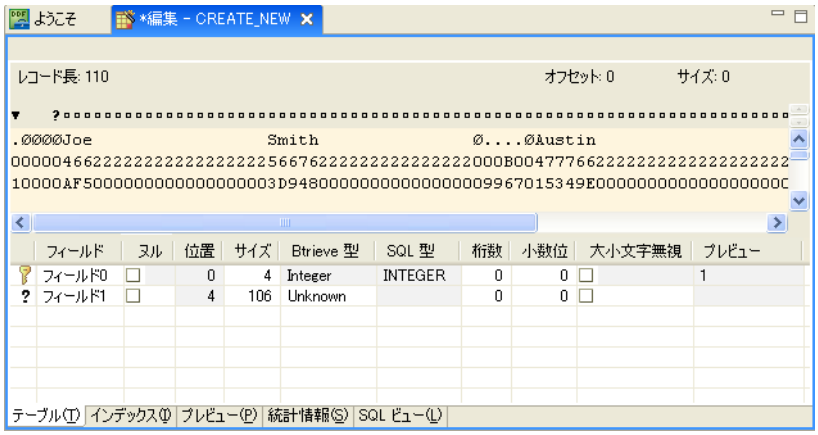
2　ファイルが含まれる CREATE_NEW.MKD フォルダーを右クリックし、［**テーブル定義の作成**］をクリックします。



テーブル定義を作成する場合、DDF Builder では新しいテーブル名とこのテーブル定義のセットを関連付けるための［テーブル定義の新規作成］ダイアログを表示します。このテーブルはトランザクショナル ファイルに対応するリレーショナル テーブルと考えてください。テーブル定義の作成が終了すると、SQL テーブルは元の Btrieve ファイルと同じ構造を正確に反映しているはずです。

3　テーブル名としてデフォルトの「CREATE_NEW」を使用し、[OK]をクリックします。

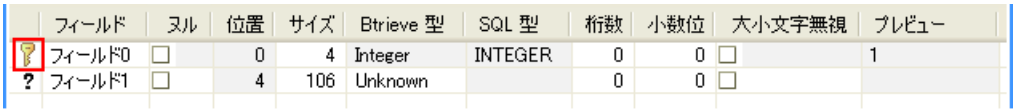
テーブルの名前を指定して[OK]をクリックすると、DDF Builder では Btrieve ファイルを開くことができるようにするために、そのファイルの分析を開始します。DDF Builder ではファイル内に認識できるキーやインデックスがあるかを判断し、その結果をテーブル定義エディター内に表示します。テーブル定義エディターの表示は次のようになります。



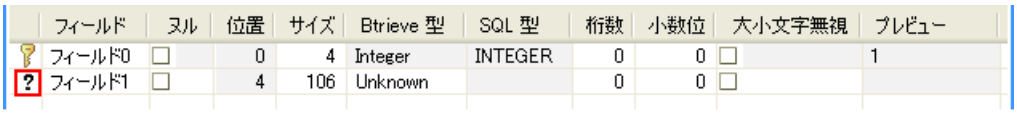
ファイルを開くことができたら、DDF Builder の調査結果をより詳しく見ていきます。

DDF Builder による調査結果を検証する

DDF Builder ではファイルの一般的な統計情報と既知のキーまたはインデックスを基にデータを分析します。DDF Builder による分析では、このファイルのレコードの先頭 4 バイトはキーであり、そのキーは Btrieve データ型で Integer であることを検出しています。このキーはテーブル定義エディターの先頭列でキー アイコンとして示されています。



レコード内のキー以外の残りの部分は DDF Builder で解読できないため 1 つのグループにまとめられ、テーブル定義エディターで疑問符アイコンとして示し Btrieve 型には不明（Unknown）が割り当てられます。



Btrieve ファイル用のテーブル定義を作成するにあたり、DDF Builder で検出された不明なフィールドを必要に応じてさまざまなフィールドへ分割するという作業が生じます。このまま先に進めば、各フィールドを詳細に定義します。

レコード フィールドの定義を始める前に、テーブル定義エディターでヌルがどのように扱われるかを理解しておいてください。

メモ このチュートリアルでは真のヌルを使用しますが、かなり古いバージョンの Zen では真のヌルをサポートしていなかったことに注意してください。サポートしていたのはレガシー ヌルのみです。真のヌルは Pervasive.SQL 2000 で初めて導入されました。Pervasive.SQL 2000 より前に作成されたファイルで作業する場合、ヌルに関する次のセクションは省いてもかまいません。

ヌルに関する注記

このチュートリアルでレコード フィールドを定義する場合、フィールドにヌル値を許可するかどうかについて留意することが重要です。なぜなら、レコードの一部分でヌル値を許可する場合、1 バイトが余分にフィールドへ追加されるからです。Btrieve ファイルでは、ヌル値が許可されるレコード部分はヌル インジケーター バイトによって指定されます。ヌル インジケーター バイトは、テーブル定義エディター ページの未加工データ ビューでフィールドまたは列の直前のバイトに該当します。［ヌル］チェック ボックスが選択されると、ヌル インジケーター バイトがアクティブになり、そのヌル インジケーター バイト分を確保するためフィールドのサイズは自動的に 1 バイト削減されます。

サイズが 50 バイトでヌル値を許可するフィールドを作成するとします。ヌル インジケーター バイトを確保するためにはフィールドのサイズを 51 バイトとして選択します。フィールドにヌル値を許可すると、そのサイズは自動的に 50 バイトに削減されます。

この情報はレコードのフィールドを定義する際に役立ちます。

メモ DDF Builder でヌルを扱う作業の詳細については、「DDF Builder におけるレガシー ヌル」および「DDF Builder における真のヌル」を参照してください。

レコード フィールドを定義する

既にご覧になったように、Btrieve ファイルを開くと DDF Builder では 4 バイトのインデックスとそれ以外を 106 バイトの不明なフィールドと判断します。不明なフィールドをレコード フィールドに定義します。

▶▶ レコード フィールドを定義するには

DDF Builder は先頭フィールドのほとんどの属性を判断しましたが、フィールド名は判断できませんでした。このため、まずは先頭フィールドに名前を付けてください。

フィールドに名前を付ける

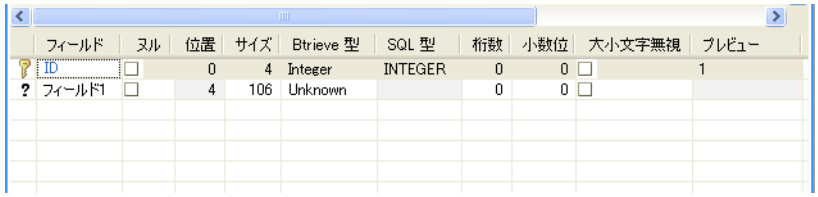
1 グリッド データ ビューで、" フィールド 0" の行を選択し、以下の情報を反映するために必要な値を入力します。

- フィールド：ID
- ヌル：チェックしない
- 位置：0
- サイズ：4
- Btrieve 型：Integer
- SQL 型：INTEGER
- 桁数：0
- 小数位：0

- 大小文字無視：チェックしない

DDF Builder ではこの行のほとんどの属性を判断しているので、変更が必要な値のみ入力してください。この行の値をすべて確認してください。

グリッド データ ビューの画面では、"ID" という名前を付けたキー行と DDF Builder で検出された " フィールド 1" という 2 つの行が示されます。



これで不明なデータを解析して定義する準備ができました。

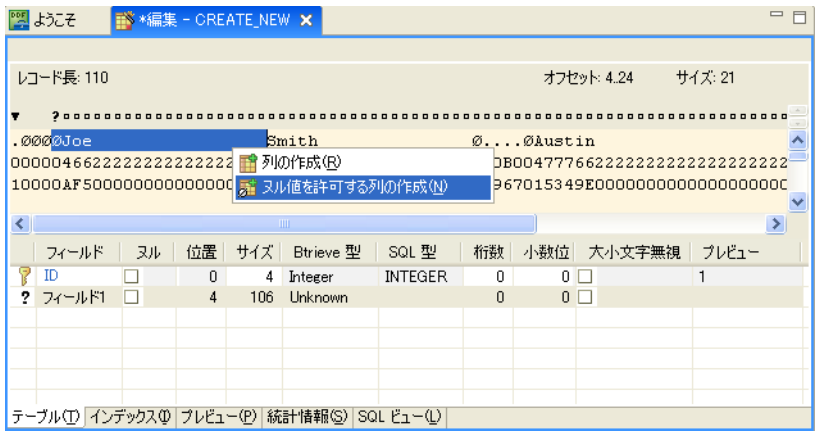
次のフィールドはヌル値を許可するよう定義します。次の手順では未加工データ ビューでデータを選択し、そのデータからヌル値を許可する列を作成する方法について説明します。

未加工データ ビューでヌル値を許可する列を作成する

2 未加工データ ビューでオフセット 4 から 21 バイト分のデータを選択します。

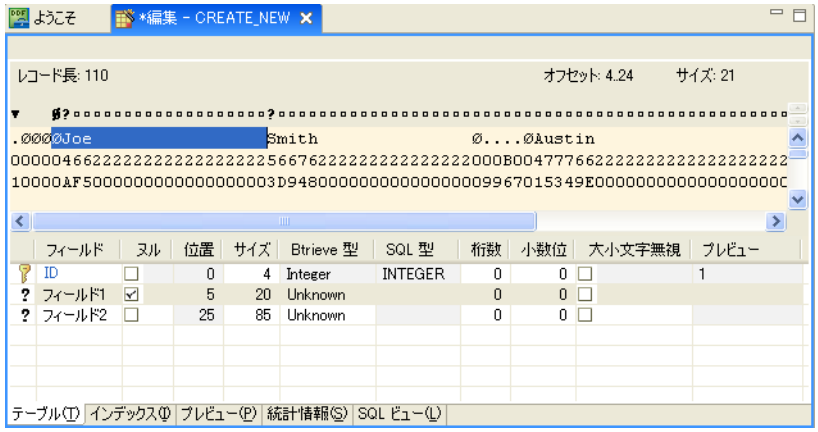
ヒント オフセット 4 を特定するには、未加工データ ビューで先頭レコードのすぐ上にある疑問符マークを見つけるか、［オフセット］インジケーターの値が 4 になる場所にカーソルを置きます。

3 右クリックして［**ヌル値を許可する列の作成**］を選択します。



ヒント 列を 1 つ作成すると、未加工データ ビューとグリッド データ ビューのどちらにも、レコードの残りのバイトをまとめた新しい列が自動的に作成されます。

21 バイト分を選択したはずですが、グリッド データ ビューでそのサイズが 20 バイトと表示されることに注目してください。これは前に説明したヌル インジケーターによるものです。ヌル インジケーター バイトは未加工データ ビューに § 記号で示され、追加バイトとして割り当てられます。



エディターには、4 バイトの整数値（ID）、前の手順で作成したヌル値を許可する列（フィールド 1）、そして新たに残りのバイトをまとめた不明なフィールド（フィールド 2）という 3 つのフィールドが表示されます。

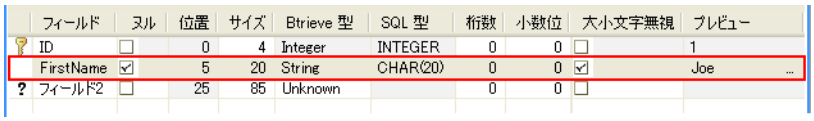
作業を進める前に、ヌル値を許可する列の定義を済ませてください。

4 グリッド データ ビューで " フィールド 1" の行を選択します。

5 以下の情報を反映するために必要な値を入力します。

- フィールド：FirstName
- ヌル：チェックする
- 位置：5
- サイズ：20
- Btrieve 型：String
- SQL 型：CHAR(20)
- 桁数：0
- 小数位：0
- 大小文字無視：チェックする

ヒント 未加工データ ビューで選択したバイト数と、この列をヌル値を許可する列として作成していることから、サイズとヌルの選択はあらかじめ決定しています。また、このサイズでは列に対して選択できるデータ型が限定されます。



Btrieve 型を設定してそのフィールドが定義されると、それ以降、グリッド データ ビューには不明なフィールド インジケーターが表示されなくなるので注意してください。

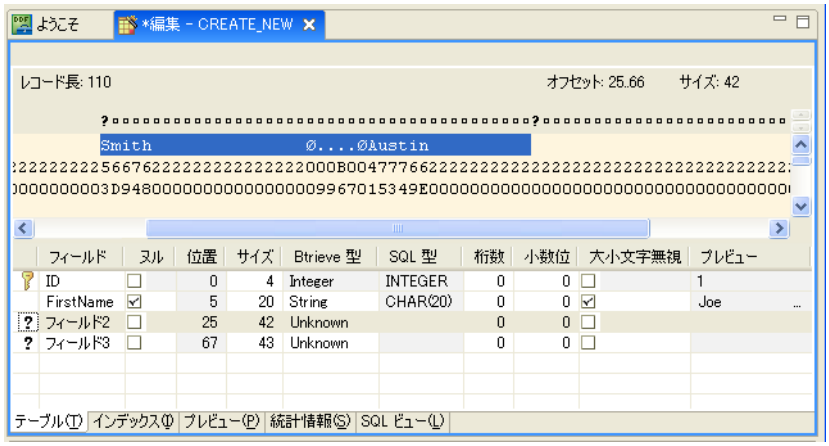
引き続き、グリッド データ ビュー内で不明な行を複数の列に分割して、ファイル内の不明なフィールドを定義します。

グリッド データ ビューで列を分割する

6 グリッド データ ビューで " フィールド 2" の行を選択します。

7 右クリックして［**列の分割**］を選択します。

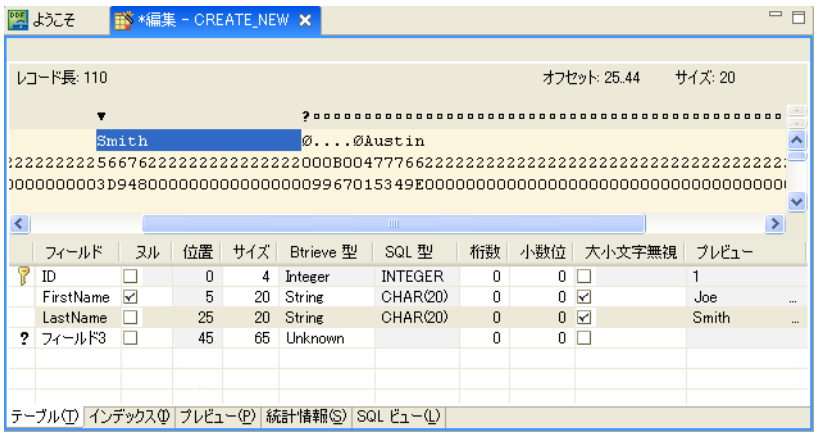
列を分割する場合、分割される 2 つの列のサイズはそれぞれ同等か、ほぼ同等になるようにします。これは、85 バイトの列を 2 つの列に分割した場合、一方の列は 42 バイトでもう一方の列は 43 バイトになるということです。



8 " フィールド 2" の行を選択します。

9 以下の情報を反映するために必要な値を入力します。

- フィールド：LastName
- ヌル：チェックしない
- 位置：25
- サイズ：20
- Btrieve 型：String
- SQL 型：CHAR(20)
- 桁数：0
- 小数位：0
- 大小文字無視：チェックする

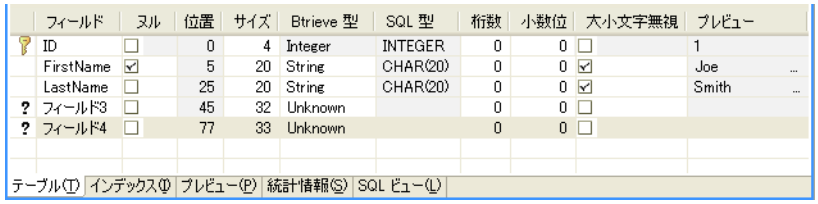


サイズを 42 から 20 に変更すると、DDF Builder では " フィールド 3" 行に余分の 22 バイトを追加します。

SQL プレビュー列には定義されたデータが表示されていることがわかります。ここでは、テーブル定義に基づき DDF Builder がデータがどのように解釈したかをプレビューしています。

10 " フィールド 3" の行を右クリックして［**列の分割**］を選択し、この行を分割します。

この列を分割すると、フィールド 3 は 32 バイトに変更され、33 バイトのフィールド 4 が作成されます。



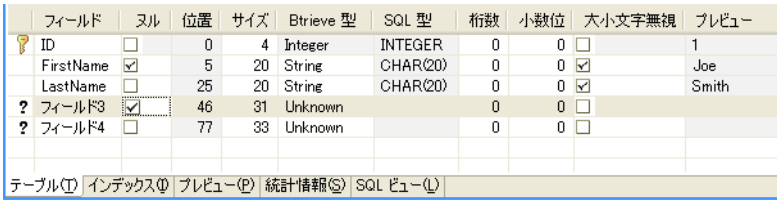
	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
	FirstName	☑	5	20	String	CHAR(20)	0	0	☑	Joe
	LastName	☐	25	20	String	CHAR(20)	0	0	☑	Smith
?	フィールド3	☐	45	32	Unknown		0	0	☐	
?	フィールド4	☐	77	33	Unknown		0	0	☐	

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

フィールド 3 を定義します。このフィールドにはヌル値を許可します。

11 ［**ヌル**］チェック ボックスを選択して、フィールドの先頭をヌル インジケーター バイトで予約します。

［ヌル］チェックボックスを選択すると、自動的にサイズが 31 に減少するので注意してください。



	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
	FirstName	☑	5	20	String	CHAR(20)	0	0	☑	Joe
	LastName	☐	25	20	String	CHAR(20)	0	0	☑	Smith
?	フィールド3	☑	46	31	Unknown		0	0	☐	
?	フィールド4	☐	77	33	Unknown		0	0	☐	

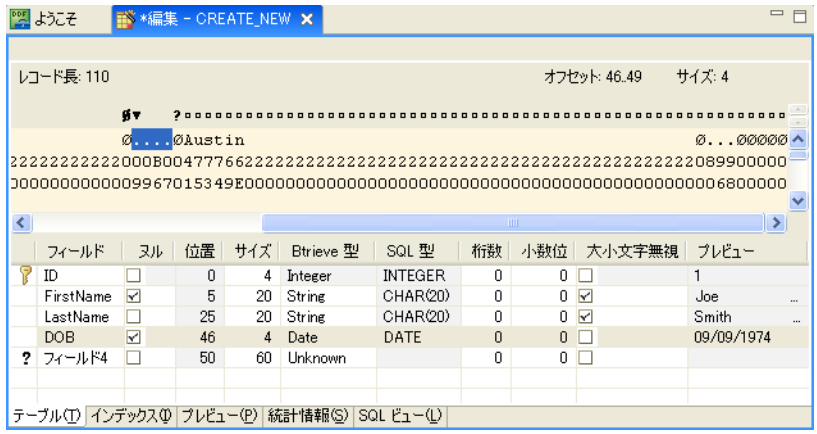
テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

12 " フィールド 3" の行を選択します。

13 以下の情報を反映するために必要な値を入力します。

- フィールド：DOB
- ヌル：チェックする
- 位置：46
- サイズ：4
- Btrieve 型：Date
- SQL 型：Date
- 桁数：0
- 小数位：0
- 大小文字無視：チェックしない

テーブル定義は次のようになります。



	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
	FirstName	☑	5	20	String	CHAR(20)	0	0	☑	Joe
	LastName	☐	25	20	String	CHAR(20)	0	0	☑	Smith
	DOB	☑	46	4	Date	DATE	0	0	☐	09/09/1974
?	フィールド4	☐	50	60	Unknown		0	0	☐	

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

サイズを 4 に変更すると、DDF Builder では残りのバイト数を次のフィールドに含めます。

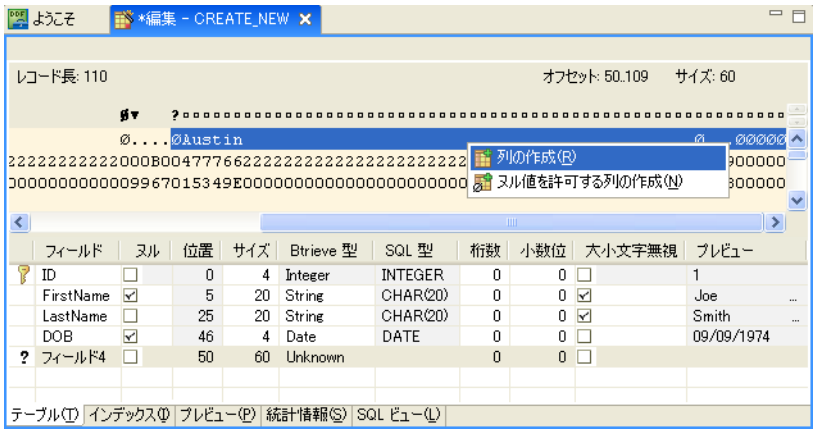
この時点で、60 バイトの不明なフィールドが残っています。データ構造ではさらに 2 つのフィールドの定義が必要であることを示しています。未加工データ ビューを使用して、その列を作成します。

未加工データ ビューから列を作成する

14 未加工データ ビューでオフセット 50 から 51 バイト分のデータを選択します。

ヒント オフセット 50 を特定するには、未加工データ ビューで先頭レコードのすぐ上にある疑問符マークを見つけるか、［オフセット］インジケーターの値が 50 になる場所にカーソルを置きます。

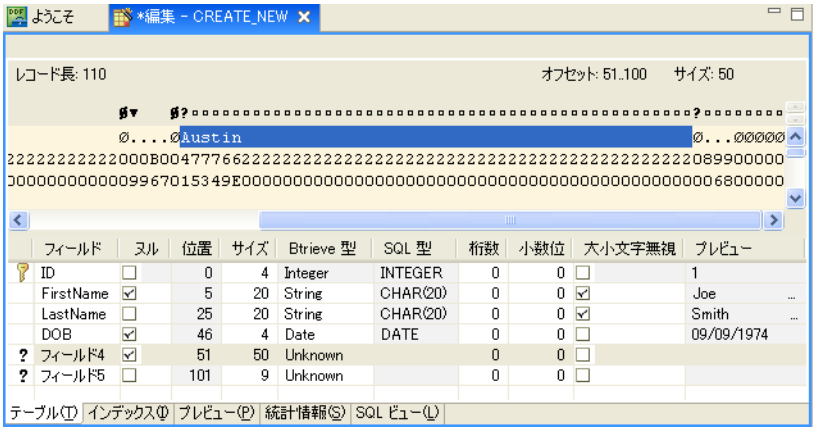
15 右クリックして［**列の作成**］を選択します。



ヌル値を許可しない列を作成しましたが、本当に必要なのはヌル値を許可する列です。グリッド データ ビューを使用して、この列をヌル値を許可する列に変更します。

グリッド データ ビューでヌル値不可の列をヌル値を許可する列に変更する

16 " フィールド 4" の行を選択して、［**ヌル**］チェックボックスを選択します。



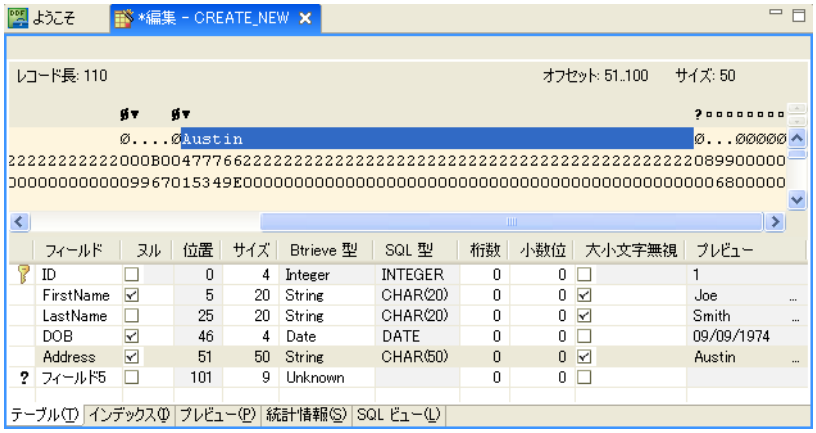
［ヌル］チェックボックスをチェックすると、ヌル バイト インジケーターを確保するのでサイズが 50 バイトに減少します。

17 この他、以下のフィールド情報を反映するために必要な値を入力します。

- フィールド：Address
- ヌル：チェックする
- 位置：51
- サイズ：50
- Btrieve 型：String

- SQL 型：CHAR(50)
- 大小文字無視：チェックする

テーブル定義は次のようになります。



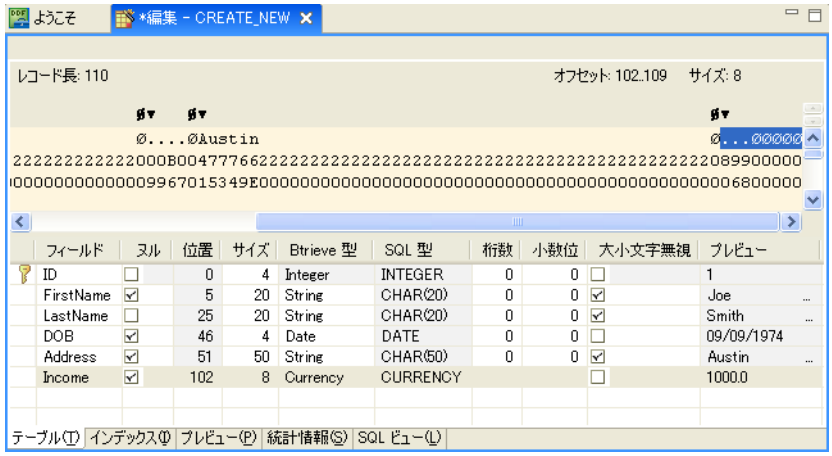
最後に、ファイルの最後のフィールドを定義します。

18 グリッド データ ビューで " フィールド 5" の行を選択し、以下の変更をこの順番で行います。

- ヌル：チェックする
- フィールド：Income
- Btrieve 型：Currency

最初に［ヌル］チェックボックスを選択することによって、サイズは自動的に修正されます。最初にサイズを 8 に変更すると、DDF Builder では最後に残った 1 バイトから不明な列を新たに作成してしまいます。

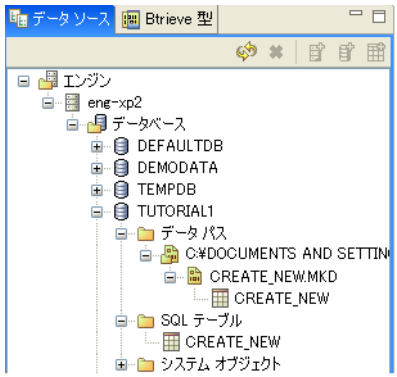
これで、ファイルのフィールドに対する定義が終了しました。テーブル定義は次のようになります。



作業を進める前に、テーブル定義を保存しておいてください。

19 ［**ファイル**］>［**保管**］または、ツールバーの［保存］アイコンをクリックして変更を保存します。

これで DDF にテーブル定義を作成することができました。作業を保存すると、データ ソース エクスプローラーで［SQL テーブル］ノードの下にその SQL テーブルが作成されます。

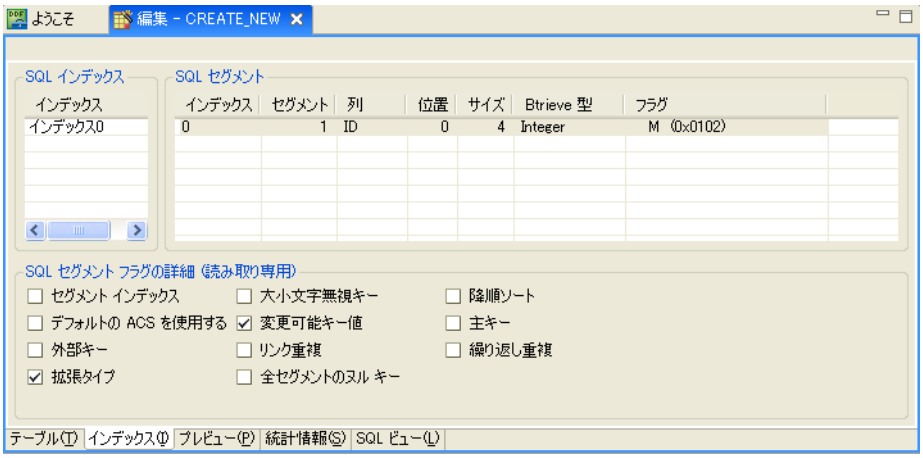


テーブル定義を作成したので、インデックスを見てみましょう。

インデックス情報を見直す

DDF Builder がファイルで検出したインデックスを見てみましょう。

1　テーブル定義エディターで［**インデックス**］タブをクリックします。



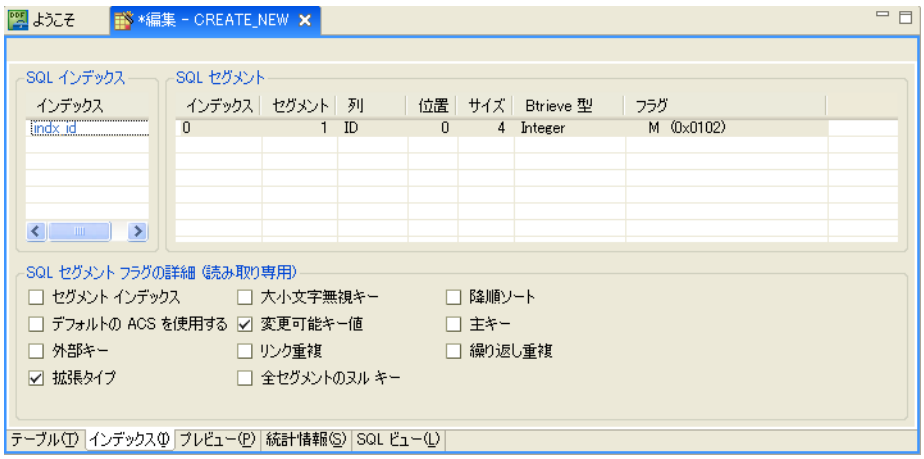
DDF Builder ではファイルに作成されているインデックスを検出することができますが、Btrieve はインデックス名を使用しないため、DDF Builder でインデックスに関連付ける名前はありません。

［インデックス］ページは主に読み取り専用ですが、インデックス名については変更可能です。

インデックスの名前を付ける

2　［インデックス］列の " インデックス 0" エントリをダブルクリックして選択します。

3　データ構造から、インデックス名として「indx_id」と入力します。



4 作業を進める前に、インデックスの変更を保存しておいてください。［**ファイル**］>［**保管**］をクリックして保存します。

DDF Builder でインデックスに対して行うことができるのはこれだけです。

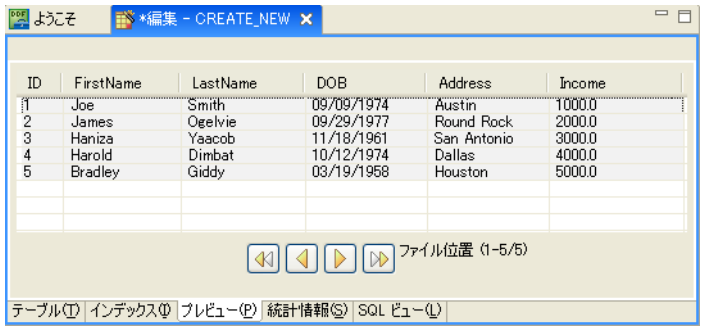
テーブル定義を終えたらそのデータを必ずプレビューするようにしてください。

定義したデータをプレビューする

次に、作成したテーブル定義用のファイルにあるデータを再度確認します。

1 DDF Builder が起動していれば、テーブル定義エディターで［**プレビュー**］タブをクリックします。

［プレビュー］ページは次のようになっているはずです。



［プレビュー］ページでは、作成したテーブル定義を使用して書式設定されたファイルのデータを見ることができます。見てわかるように、すべてのデータが列名やデータ型に応じて適正に表示されています。

このページの下部にあるボタンを使用すると、データを移動して全レコードを検証することができます。場合によっては予期しない書式のデータが見つかり、テーブル定義を完成させるために対処する必要があるかもしれません。

予測どおりに表示されていないデータが見つかった場合は、次のセクションの「チュートリアル 2 ― DDF Builder でテーブル定義 – 変更する」を参照してください。

既に保存済みのテーブル定義に対して変更を行う場合、DDF Builder を閉じる前には作業を必ず保存してください。

終わりに

おめでとうございます。これでチュートリアル 1 は完了し、リレーショナル アクセスを使用して Btrieve ファイルのデータにアクセスできる DDF のセットを持つことができました。

次のチュートリアルでは Btrieve ファイル用に作成されている DDF のセットに対する変更に取り組みます。また、DDF Builder で自動的に検出される DDF に伴ういくつかの問題についても取り上げます。

ヒント このチュートリアルではデータベースと DSN を新規作成しました。システムからこれらを削除する必要がある場合は、この時点で削除してもかまいません。

チュートリアル 2 ― DDF Builder でテーブル定義 – 変更する

古い DDF または不正な DDF を処理する場合にはさまざまなシナリオが発生するので、このチュートリアルではそれぞれのシナリオについて余すところなく取り上げています。そのため、このチュートリアルはさまざまなレッスンで構成され、Btrieve ファイルや既存のテーブル定義で作業するときに発生する可能性のある特定の状況をレッスンごとに詳しく述べています。各レッスンは、混乱を招かないよう 1 つの問題に専念して解決できるよう設計されています。

このチュートリアルは、処理しやすい単位で分割した課題をレッスンごとに取り組むことができるよう整理されています。また、このチュートリアルはトラブルシューティング調査として参照し、発生する可能性がある状況を見つけるということにも利用できます。レッスンの一覧から、探している状況について取り上げているレッスンに進み、有効な解決策を調べることができます。

シナリオ

使用可能な DDF を持つ Btrieve データ ファイルの一群があります。しかし、その DDF のテーブル定義は古いもので変更する必要があります。これはベンダーから古いファイルを受け取った、あるいはアプリケーションが変更されても DDF が適切に更新されていなかったという状況が考えられます。

目的

このチュートリアルの目的は、すべての Btrieve ファイルと既存のテーブル定義を開くことです。DDF Builder の調査結果を検証して必要であれば変更を行い、DDF Builder でそのテーブル定義を保存します。このテーブル定義はデータベース スキーマをミラー化するので、Btrieve ファイルのデータを厳密に表示できます。

このチュートリアルの各レッスン内容は次のとおりです。

「レッスン 1 ― v3.00 の DDF を使った作業」

「レッスン 2 ― v6.x より前のファイル形式での作業」

「レッスン 3 ― 不正なデータ型とサイズ」

「レッスン 4 – 列の定義の重複」

「レッスン 5 ― ファイル / フィールド フラグの不一致」

「レッスン 6 – インデックスの不一致」

「レッスン 7 – 可変長レコードの不一致」

「レッスン 8 – レコード長の不一致」

レッスン 1 ー v3.00 の DDF を使った作業

シナリオ

このレッスンでは、既存のテーブル定義セットを持つ 1 つのデータベースがあります。このテーブル定義はかなり古いバージョンの Zen で作成されたもので、今回リリースされた DDF Builder でサポートされる Pervasive.SQL 2000 とはもう互換性がありません。

注意 DDF Builder ではバージョン 4.xx の DDF をサポートしますが、Scalable SQL バージョン 4.xx より前に作成された DDF はサポートしません。

目的

このレッスンの目的は、DDF Builder でその DDF を開くことです。このチュートリアルでは、DDF Builder でこれらのファイルがどのように処理されるかを説明し、またそのファイルを DDF Builder と互換性を持つファイルへ変換できるようにする解決策を提供します。

必要な知識

このレッスンで使用するファイルは V3 という名前のフォルダーにあります。デフォルトのインストール先にインストールしている場合、このフォルダーは次の場所にあります。

<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2¥v3

作業を続ける前に、このデータベース用の DSN を作成する必要があります。

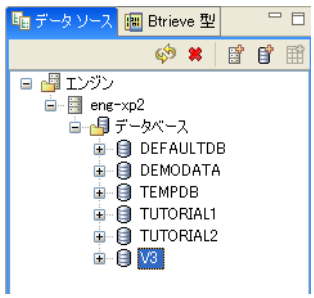
メモ DSN の作成について情報が必要な場合は、「データ ソース名（DSN）の作成」を参照してください。

Btrieve ファイルを開く

DDF Builder をまだ実行していない場合は起動します。

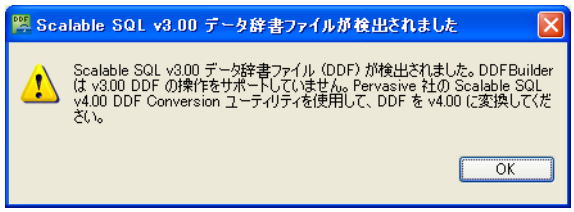
1 DDF Builder のデータ ソース エクスプローラーで、[**データベース**] ノードを展開して使用可能データベースをすべて表示します。

2 一覧から V3 データベースを見つけます。



3 V3 データベース アイコンをダブルクリックします。

次のメッセージが表示されます。



4 ［OK］をクリックしてこのメッセージ ウィンドウを閉じます。

本来、これらのファイルは、現在ではサポートされていないバージョンの製品で作成されているため開くことができません。

警告メッセージを理解する

DDF Builder では Scalable SQL v3.00 のデータ辞書ファイルを開くことができません。これはその時点（Scalable SQL 3.01）で使用されていたフォーマットが現在サポートされていないからです。

ヒント DDF Builder は PSQL v9 以上のバージョンでサポートされます。Scalable SQL v4.00 以上で作成された DDF は Btrieve データ ファイル v6.x 以上と一緒にサポートされます。

ファイルを変換する方法

古い DDF がある場合はそれらを新しいフォーマットに変換する必要があります。以下の手順を使用します。

1 既存の古いデータベースをバックアップするため、データ ファイルと DDF を別の保存場所にコピーします。

2 Zen Control Center（ZenCC）を使用して古いデータベースの全テーブルのテーブル スキーマをエクスポートします。『*Zen User's Guide*』の「データベース スキーマをエクスポートするには」を参照してください。["IN DICTIONARY" 句を CREATE ステートメントに追加する］エクスポート オプションを必ず選択しておいてください。

3 エクスポート ファイルの CREATE INDEX ステートメントで、各テーブルのインデックス セグメントが正しい順序で置かれていることを確認してください。

テーブルのインデックスは元のデータ ファイルと同じ順序になっていなければなりません。エクスポートされるテーブル スキーマはインデックスをアルファベット順に置くので、このステートメントを並べ替える必要があるかもしれません。

インデックスの順序を調べるには、元のデータ ファイルに対して butil -stat コマンドを使用します。『*Advanced Operations Guide*』の「データ ファイル情報の表示」を参照してください。また、dbo.fSQLStatistics カタログ関数を使用して調べることもできます。『*SQL Engine Reference*』の「dbo.fSQLStatistics」を参照してください。

4 一般的には、古いファイルを変換するので、次の行をエクスポート スキーマ ファイルの**先頭**行として追加する必要があります。

```
SET TRUENULLCREATE = OFF;
```

次の行を、ファイルの**最終**行として追加します。

```
SET TRUENULLCREATE = ON;
```

5 ZenCC で新しいデータベースを作成します。『*Zen User's Guide*』の「新規データベースを作成するには」を参照してください。

6 ZenCC の SQL Editor で、エクスポート スキーマ ファイルの CREATE TABLE SQL ステートメントを新しいデータベースに対して実行し、すべてのテーブルを作成します。『*Zen User's Guide*』の 「SQL スクリプトを開くには」を参照してください。

7 古いデータベースから古い Zen データ ファイル（DDF 以外）を、新しいデータベース データ ファイル用の保存場所にコピーします。

SQL を使用（ZenCC などを使用）して既存のデータにアクセスできるようになります。

8 DDF Builder を使用して、定義がなかった Btrieve ファイルに対しテーブル定義を作成し、また問題のある定義を修正します。「変換に関する注意」も参照してください。

9 必要であれば、データ ファイルをリビルドします（6.x 以前のファイル形式の場合に必要となる可能性があります）。Btrieve データ ファイルのリビルドの詳細については、次のセクション、「レッスン 2 ― v6.x より前のファイル形式での作業」に進んでください。

変換に関する注意

上記の手順は、シンプルなデータ ファイルに対する最も基本的な作業です。まずはこの手順を試して変換の結果を調べてください。

以下の 2 つの状況がある場合は、さらに作業が必要です。

- 「オルタネート コレーティング シーケンス」
- 「ZenCC および DDF Builder の使用を必要とするテーブル定義」

オルタネート コレーティング シーケンス

テーブルの列がオルタネート コレーティング シーケンス（ACS）を使用する場合、CREATE TABLE ステートメントを手動で変更する必要があります。ZenCC ではテーブル スキーマをエクスポートするときに ACS を含めません。

列が ACS を使用しているかどうかを調べるには、DDF Builder でテーブルの一貫性を確認します（「テーブルの一貫性のチェック」を参照）。属性と物理キーファイル セグメントが一致しないというメッセージが通知された場合、その列は ACS を使用しています。

次の例は変更済みの CREATE TABLE ステートメントで、照合順序が手動で追加されています。

```
SET TRUENULLCREATE = OFF;
CREATE TABLE "PATAPP" IN DICTIONARY USING 'PATAPP.DTA' (
    "ID" CHAR(6) NOT NULL COLLATE 'UPPER.alt',
    "Appointment Date" DATE NOT NULL,
    "Appointment Time" TIME NOT NULL,
    "AMPM" CHAR(4) NOT NULL COLLATE 'UPPER.alt',
    "Doctor" CHAR(12) NOT NULL,
    "Code" CHAR(3) NOT NULL COLLATE 'UPPER.alt',
    "Amount Paid" MONEY(14,2) NOT NULL,
    "Date Paid" DATE NOT NULL);
CREATE INDEX "index_0" IN DICTIONARY ON "PATAPP" (
    "Appointment Date" ,
    "AMPM" ,
    "Appointment Time" );
CREATE INDEX "index_1" IN DICTIONARY ON "PATAPP" (
    "ID" );
CREATE INDEX "index_2" IN DICTIONARY ON "PATAPP" (
    "Code" );
SET TRUENULLCREATE = ON;
```

また、インデックスの順序がテーブルに対して正しくない場合は、CREATE INDEX ステートメントを並び替えて正しい順序でインデックス セグメントを置く必要があります。

ZenCC および DDF Builder の使用を必要とするテーブル定義

テーブル定義によっては、解決が困難なため ZenCC と DDF Builder の両方を使用する必要がある場合もあります。列が DDF Builder で " 不明 " と示された場合は、ZenCC の Table Editor でその列が含まれるテーブルを表示し、DDF

Builder 側でその列を同様に定義する必要があります。両方のユーティリティを用いて作業することにより、テーブル定義を完了させることができます。

終わりに

このレッスンでは、DDF Builder で Scalable SQL v3.01 の DDF がどのように処理されるかをご紹介し、またそれら DDF を本ソフトウェアの現行バージョンで作業できるように変換するための解決策も提供しました。

また、Scalable SQL v3.01 の DDF で作成されたデータベースには、v6.x より前のファイル形式の Btrieve データファイルが含まれている可能性があるので、それらのファイルも DDF Builder を使用する前にリビルドしておく必要があることも学習しました。

レッスン 2 ー v6.x より前のファイル形式での作業

シナリオ

このレッスンでは、既存のテーブル定義セットを持つ Btrieve ファイルがあります。この Btrieve ファイルは v6.x より前のファイル形式で、本リリースの DDF Builder でサポートされている Zen とはもう互換性がありません。

注意 DDF Builder ではバージョン 6.x のファイルをサポートしますが、バージョン 6.x より前に作成されたファイルはサポートしません。

目的

このレッスンの目的は、DDF Builder でそのファイルを開くことです。ここでは、DDF Builder で v6.x より前のファイル形式と v6.x 以上のファイル形式がどのように処理されるかを説明します。また、v6.x より前の Btrieve ファイルを DDF Builder に対応させるための解決策も提供します。

必要な知識

このレッスンで使用するファイル、および残りのレッスンで使用するファイルは Tutorial2 というフォルダーにあります。デフォルトのインストール先にインストールしている場合、このフォルダーは次の場所にあります。

<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2

作業を続ける前に、このデータベース用の DSN を作成する必要があります。

このレッスンで作業対象となるファイルは KO.BTR と KO.MKD です。

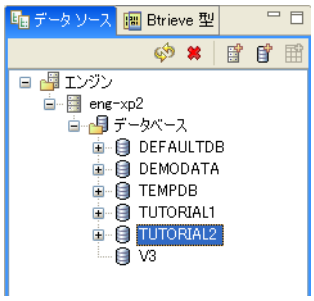
メモ DSN の作成について情報が必要な場合は、「データ ソース名（DSN）の作成」を参照してください。

Btrieve ファイルを開く

DDF Builder をまだ実行していない場合は起動します。

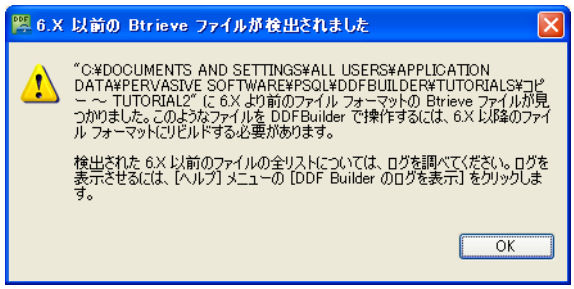
1 DDF Builder のデータ ソース エクスプローラーで、[データベース] ノードを展開して使用可能データベースをすべて表示します。

2 一覧から Tutorial2 データベースを見つけます。



3 Tutorial2 データベース アイコンをダブルクリックします。

次のメッセージが表示されます。



4 ［OK］をクリックしてこのメッセージ ウィンドウを閉じます。

フォルダー内にあるいくつかのファイルが v6.x より前の形式であったため、全ファイルを開くことができませんでした。

警告メッセージを理解する

DDF Builder ではバージョン 5.x 以下の形式のファイルが完全にサポートされなくなったので、そのファイルは開くことができません。v6.x より前の形式のファイルを使用するには、そのファイルをバージョン 6.x 以上のファイル形式にリビルドする必要があります。

ヒント DDF Builder は PSQL v9 以上でサポートされます。

バージョン 5.x 以下のファイルをリビルドする前にログ ファイルを確認し、リビルドする必要があるファイルを特定してください。

ログ ファイルを表示する

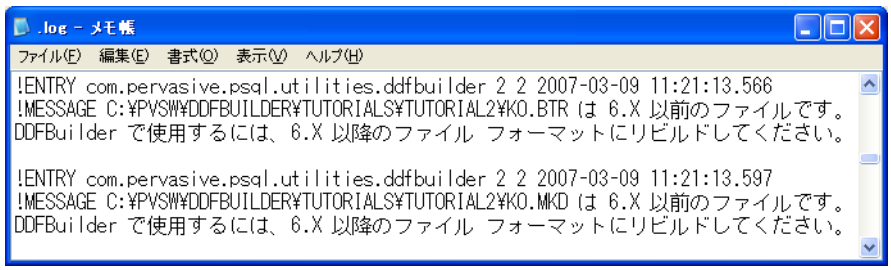
リビルドする必要があるファイルは DDF Builder ログ ファイルに記録されています。

▶▶ ログ ファイルにアクセスするには

1 DDF Builder のメニュー バーで［**ヘルプ**］をクリックします。

2 ［**DDF Builder のログを表示**］をクリックします。

ログ ファイルは、ご使用のシステムでデフォルト設定されているテキスト エディターに表示されます。次の画面は、このレッスン用のログ ファイルのサンプルです。



ログ ファイルには v6.x より前の形式のファイルがリストアップされ、そのファイルの完全なデータ パスも記録されています。このデータ パスを使用して、リビルドが必要なファイルの場所を見つけることができます。

ファイルをリビルドする方法

Zen Rebuild ユーティリティを使用して、ファイルをバージョン 6.x 以上にリビルドします。Rebuild ユーティリティは Zen における中心的なユーティリティの 1 つとしてインストールされるもので、Zen Control Center の［ツール］メニューから使用することができます。

次に行うこと

Rebuild ユーティリティは Zen Control Center から使用できます。このユーティリティの使用法については、『*Advanced Operations Guide*』の「データ ファイルの変換」の章を参照してください。

終わりに

このレッスンでは、DDF Builder で 6.x より前のファイル形式がどのように処理されるかをご紹介し、またそれらを DDF Builder で作業できるようにリビルドするための解決策も提供しました。

レッスン 3 ー 不正なデータ型とサイズ

シナリオ

このレッスンでは、不正なデータ型とサイズを持つ Btrieve ファイルがあります。この結果、データは理解できるような書式に設定されません。

目的

このレッスンの目的は、DDF Builder で Btrieve ファイルの既存のテーブル定義を開くことです。Btrieve 型ビューですべてのデータ型とサイズを調べて必要な変更を行い、フィールドのサイズに対して的確なデータ型でデータが書式設定されるようにします。

必要な知識

このレッスンでは、TYPE_SIZE.MKD というファイルを使用します。このファイルは Tutorial2 フォルダー内にあります。デフォルトのインストール先にインストールしている場合、このフォルダーは次の場所にあります。

<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2

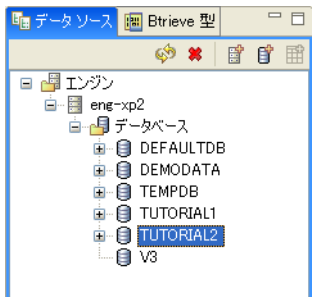
このフォルダーは Tutorial2 データベースの一部です。

メモ このチュートリアルのデータにアクセスするためには、このデータベースを指し示す DSN（データ ソース名）が必要です。まだ DSN を作成していない場合は、「データ ソース名 (DSN) の作成」を参照してください。

Btrieve ファイルを開く

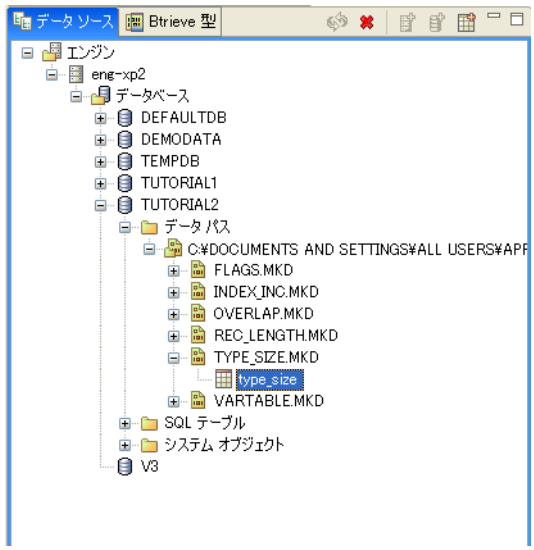
すぐ前のレッスンから引き続き作業していれば DDF Builder は起動しているはずです。終了していた場合は、DDF Builder を起動してください。

1 DDF Builder のデータ ソース エクスプローラーで、［データベース］ノードを展開し、一覧から Tutorial2 データベースを見つけます。



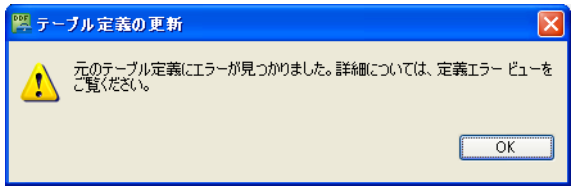
2 TUTORIAL2 データベース アイコンをダブルクリックしてそのノードを展開します。

3 ノードをさらに展開して TYPE_SIZE.MKD ファイルを探し、それに関連付けられている SQL テーブルを表示します。



4 type_size SQL テーブル名をダブルクリックするか、右クリックして［**テーブル定義の編集**］をクリックします。

テーブル定義エディターでは、次のメッセージが表示されます。



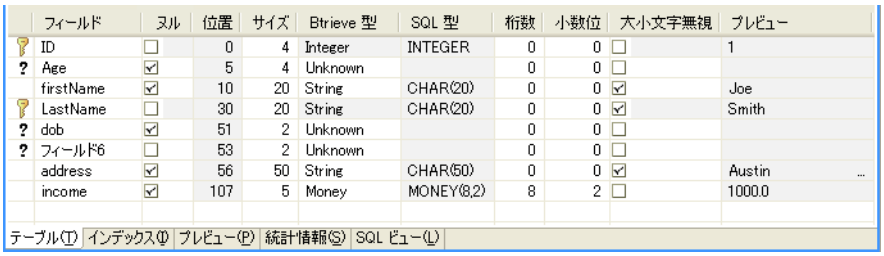
このメッセージは、DDF Builder で既存のテーブル定義を分析したら、その定義に問題があったということを知らせています。この結果、DDF Builder で既存のテーブル定義を開いて表示するために、いくつか修正を行う必要がありました。

5 ［OK］をクリックすると、このメッセージがクリアされテーブル定義が表示されます。

ヒント 発生する可能性がある定義エラーの詳細については、「テーブル定義エラー」を参照してください。

不一致を探す

まずは DDF Builder インターフェイスで処理対象の不一致を確認します。特に、グリッド データ ビューと定義エラー ビューを見てください。



	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
🔑	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
?	Age	☑	5	4	Unknown		0	0	☐	
	firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
🔑	LastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
?	dob	☑	51	2	Unknown		0	0	☐	
?	フィールド6	☐	53	2	Unknown		0	0	☐	
	address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
	income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

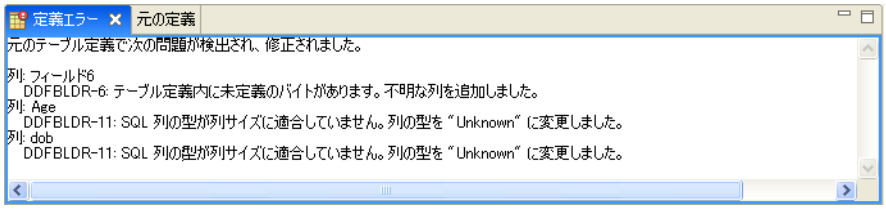
テーブル定義エディターのグリッド データ ビューでは、既存のテーブル定義に対して DDF Builder が行った変更を反映したテーブル定義が表示されています。

メモ DDF Builder が行った変更は自動的に保存されません。DDF Builder による修正は保存する必要があります。

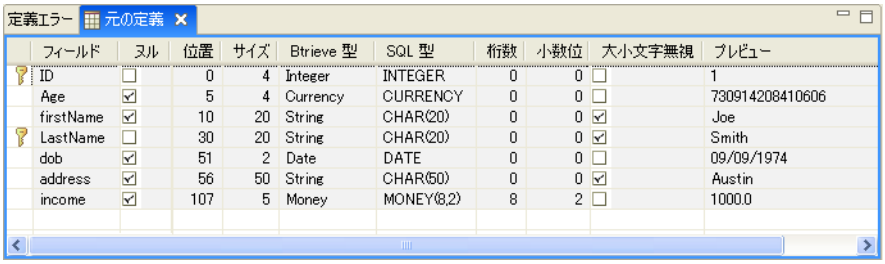
この例で、DDF Builder は新たに作成したフィールドや、DDF Builder がデータ型を「不明」に変更したフィールドに対して不明な列インジケーターを追加することにより、注目すべき列を視覚的に示しています。

ヒント グリッド データ ビューにおける属性の詳細については、「グリッド データ ビューでのフィールド属性」を参照してください。

定義エラー ビューでは DDF Builder によって検出および変更された問題を表示します。



DDF Builder によって変更が行われる前の元のテーブル定義は、元の定義ビューで見ることができます。



エラーについて理解する

定義エラー ビューでは以下のことを通知します。

- 問題が発生した場所
- DDF Builder で検出された問題
- DDF Builder で元のテーブル定義に対して行った変更事項

定義エラーでは次の 3 つの問題を挙げています。

問題が発生した場所	DDF Builder で検出された問題	必要な対処
列：フィールド 6	DDFBLDR-6:テーブル定義内に未定義のバイトがあります。不明な列を追加しました。	DDF Builder が未定義のバイトを明らかにするために追加した不明な列を定義する必要があります。
列：Age	DDFBLDR-11：SQL 列の型が列サイズに適合していません。列の型を "Unknown" に変更しました。	DDF Builder が "Unknown"（不明）に変更した列の型を定義する必要があります。
列：dob	DDFBLDR-11：SQL 列の型が列サイズに適合していません。列の型を "Unknown" に変更しました。	DDF Builder が "Unknown"（不明）に変更した列の型を定義する必要があります。

1 番目のエラーは、元のテーブル定義では定義されていないバイト群がファイルに含まれていることを示しています。DDF Builder はこれら未定義のバイトを新しい列として作成し、その列に " フィールド 6" という名前を付けました。

また、このファイルには列のデータ型がその列のサイズに対して不正な 2 つのフィールド（Age と dob）があります。どちらも、フィールドの型が "Unknown"（不明）に変更されています。Age 列について詳しく見る前に、まず DDF Builder で判断できることとできないことを確認します。

不正なデータ型とサイズ

DDF Builder では、指定したデータ型に対して有効なサイズを確かめることはできませんが、指定したサイズに対して有効なデータ型を検証することはできます。DDF Builder では問題を検出して一般的な変更を行い、その変更を記録することはできますが、選択されたデータ型が正しいかどうかを判断することはできません。

DDF Builder ではフィールドのサイズに応じて選択できるデータ型が制限されます。たとえば Age 列の不正なデータ型とサイズについて見てみると、サイズが 4 バイトになっています。グリッド データ ビューの［Btrieve 型］リストには、4 バイトのサイズに対して適用可能なデータ型のみが設定されます。DDF Builder で選択可能なデータ型を 1 つずつ試してみるのではなく、Btrieve 型ビューや［プレビュー］ページでそのデータを見て適切なデータ型を判断してください。

次に、4 バイトのサイズに対して適用可能なデータ型を調べ、ファイルのデータをプレビューしてどのデータ型が最も適しているかを見ます。

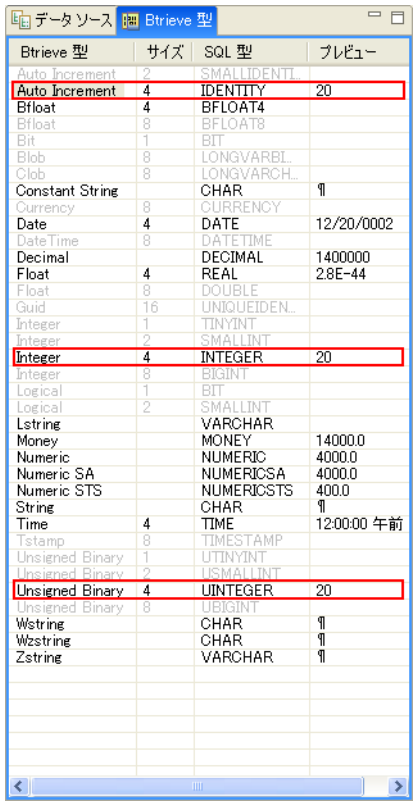
データ型とサイズを検証する

フィールド サイズに対して有効なデータ型で書式設定されたデータを見るには、テーブル定義エディターの左側にある［Btrieve 型］タブの Btrieve 型ビューを使用します。

1 テーブル定義エディターで Age 列をクリックして選択します。

ヒント テーブル定義の Age 列を選択すると、次で説明しているように［Btrieve 型］タブで特定のデータを見ることができます。

2 DDF Builder インターフェイスの左側にある［Btrieve 型］タブをクリックを表示します。［Btrieve 型］タブは次のように表示されます。

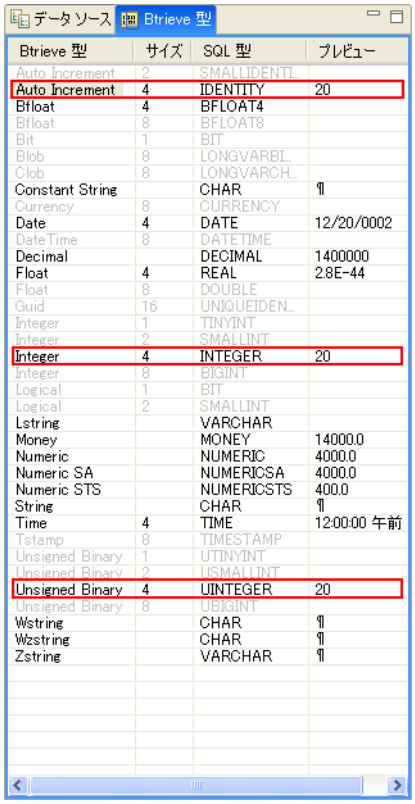


ヒント Btrieve 型ビューの詳細については、「Btrieve 型」を参照してください。

3 Btrieve 型ビューの［プレビュー］列に表示されるデータを注意深く見てください。

Btrieve 型ビューでは、サイズが 4 バイトの場合に設定可能なデータ型のみを表示します。このように表示することで、その他の不正なデータ型をすべて除外することができるため非常に役に立ちます。

その列が年齢を表すものでサイズが 4 バイトであることを知っていれば、そのデータに対して妥当なのは 3 種類のデータ型のみであることがわかります。



次に、Auto Increment を妥当なデータ型の対象から除外します。これは Auto Increment データ型を使用すると新しいレコードごとに値が自動的に 1 ずつ増えるためです。自分の年齢に 1 を足す人はいないでしょう。

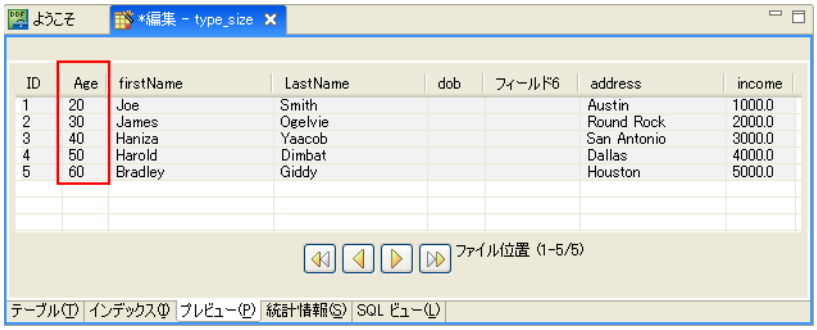
この除外によって Btrieve 型の一覧の中から Integer と Unsigned Binary という 2 つのデータ型に絞り込まれました。

Integer および Unsigned Binary データ型は実際によく似ています。これらのデータ型を試してみて、データがどのように解釈されるかを確認します。まず、Age 列の Btrieve 型として Integer を選択します。

4 ［Btrieve 型］リストから "Integer" を選択します。

グリッド データ ビューの［プレビュー］列では、データが適切に受け入れられているように見えます。［プレビュー］ページでファイルのすべてのデータを見るようにしてください。

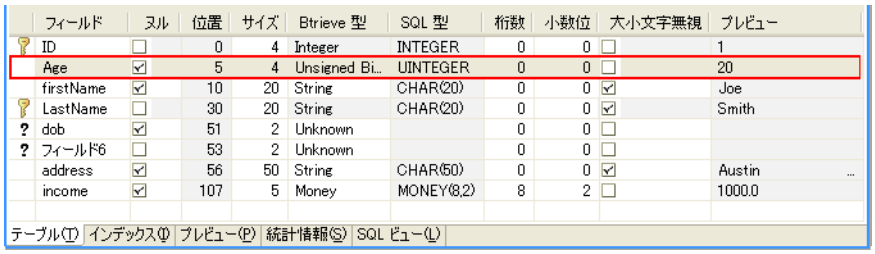
5 テーブル定義エディターで［**プレビュー**］タブをクリックします。ファイルの全データを読みやすいレイアウトで見ることができます。



ファイルの全データを見ることによって全レコードをすばやく調査できるので、不正なデータ型かどうかをより確実に見極めるのに役立ちます。

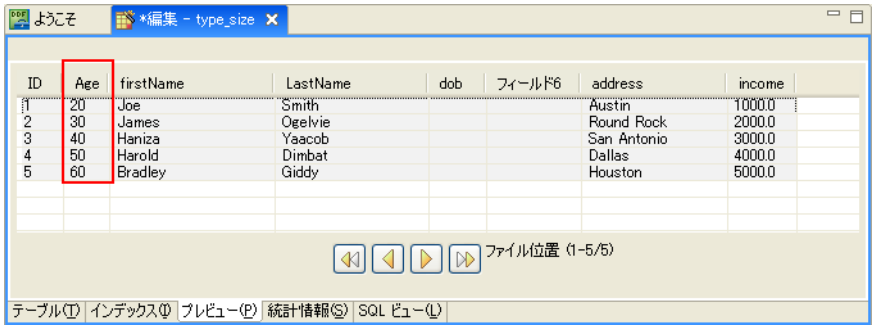
Integer データ型を使用すると、データが適切に書式設定されているように見えることが確認できます。念のため、Unsigned Binary データ型も使用して同じように検証してください。

6 ［Btrieve 型］リストから "Unsigned Binary" を選択します。



同様に、グリッド データ ビューの［プレビュー］列には適切な書式でデータが表示されています。確認のため、Integer データ型を使用したときと同じように、［プレビュー］ページでファイルの全データを見ます。

7 テーブル定義エディターでで［**プレビュー**］タブをクリックして、ファイルの全データを読みやすいレイアウトで表示します。



Integer および Unsigned Binary データ型はいずれもデータを適切に受け入れて理解できるように書式設定します。また、どちらのデータ型も 1、2、4 および 8 バイトをサポートします。このため、どちらかのデータ型を選択するための決定的な理由はないように思われます。この例は、テーブル定義を作成または変更する場合には、データの基本的な構造を知っておく必要があることを端的に示しています。

最終的な変更を行う

有効なデータ型と書式設定されたデータを再検討し、Age 列には Integer データ型を選択することにします。

	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
	Age	☑	5	4	Integer	INTEGER	0	0	☐	20
	firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
	LastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
?	dob	☑	51	2	Unknown		0	0	☐	
?	フィールド6	☐	53	2	Unknown		0	0	☐	
	address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
	income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

これで 1 番目のテーブル定義エラーが解決されました。しかし、ほかのエラーを解決するまではテーブル定義を保存することができません。

ヒント データ型が "Unknown"（不明）になっているものをすべて解決し、レコード内の未定義バイトを明確に割り当てるまで、テーブル定義を保存することはできません。

次のエラー対象の 2 つのフィールドは並んでいるので集合的に見ることができます。残りの 2 つのテーブル定義エラーを解決するには、これらを単一のエラーとして考える必要があります。

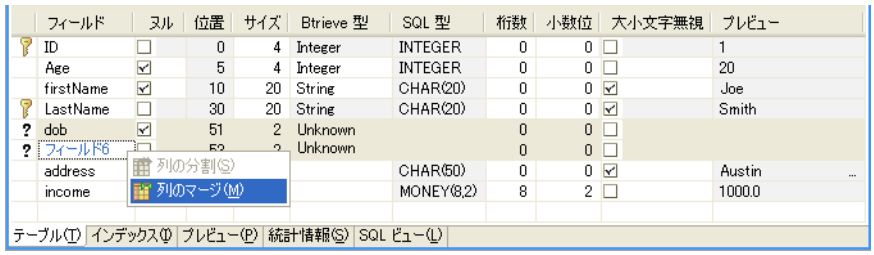
まず、dob フィールドに定義されていたデータ型（Date）は 2 バイトの列サイズに対して有効ではない、という点に DDF Builder が注目したことがわかります。その結果、DDF Builder はこのデータ型を "Unknown"（不明）に変更しました。

次に、元のテーブル定義では定義されていない 2 バイトがあります。DDF Builder はこれらの 2 バイトから新しい列（フィールド 6）を作成し、"Unknown" データ型を割り当てました。

dob フィールドの用途に適したデータ型は Date ですが、この Date データ型は 4 バイトのフィールドが必要です。"dob" 列 " と " フィールド 6" 列をマージすれば、Date データ型に適合する 4 バイトのフィールドを作成できます。2 つの列をマージして Date データ型を使用した場合、1 番目のエラーでも行ったように、そのデータが適切に受け入れられ理解できるような書式になっているかどうかを検証する必要があります。

1 テーブル定義エディターで、"dob" 列と " フィールド 6" 列の両方を（Shift キーを使って）選択します。

2 右クリックして［**列のマージ**］を選択します。



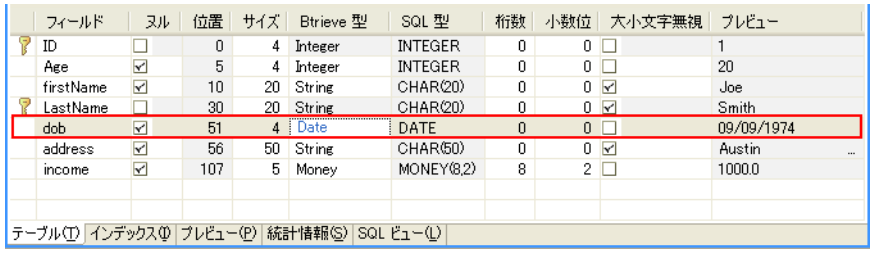
	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
	Age	☑	5	4	Integer	INTEGER	0	0	☐	20
	firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
	LastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
?	dob	☑	51	2	Unknown		0	0	☐	
?	フィールド6	☐			Unknown		0	0	☐	
	address	☑				CHAR(50)	0	0	☑	Austin
	income	☑				MONEY(8,2)	8	2	☐	1000.0

列の分割(S)
列のマージ(M)

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

2 つの列が 1 つにマージされ、4 バイトの列が作成されます。

3 ［Btrieve 型］リストから "Date" を選択します。



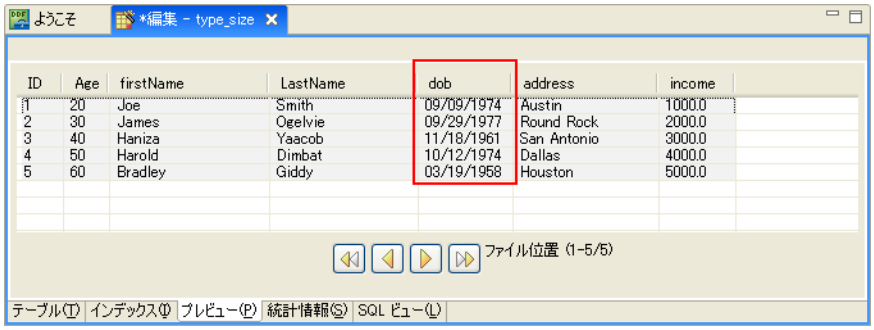
	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
	Age	☑	5	4	Integer	INTEGER	0	0	☐	20
	firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
	LastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
	dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
	address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
	income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

この情報の入力後に残りの列フィールドを見てみると、Date データ型を追加したことによってレコードのデータが適切に書式設定されることがわかります。

［プレビュー］ページで、Date データ型で書式設定されたデータがどのように処理されるかを確認することをお勧めします。

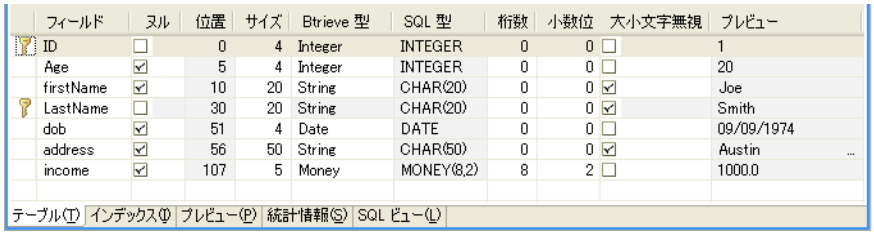
4　グリッド データ ビューの下部にある［**プレビュー**］タブをクリックします。



［プレビュー］ページの dob 列を見て、全データが適切に受け入れられ理解できる書式で設定されていることを確認できます。

テーブル定義を保存する

これでテーブル定義が完了したので、この作業を保存して変更を有効にしてください。作業を保存する前に、もう一度テーブル定義を見直してください。データ グリッド ビューは次のようになります。



Btrieve 型がすべて定義されていること、未定義のフィールドが残っていないこと、またすべてのバイトが明確に割り当てられていることが一目でわかります。これでテーブル定義を保存できます。

1　メニューから、［**ファイル**］>［**保管**］を選択します。

終わりに

このレッスンでは、データ型とサイズが適合しない場合に DDF Builder でどのように処理されるかをご紹介しました。不正なデータ型やサイズが既存のテーブル定義内でどのように表示されるかを示し、サイズとデータ型が適合して情報が適切に正しく表示できるようテーブル定義を変更するための解決例をいくつか提供しました。

レッスン 4 – 列の定義の重複

シナリオ

このレッスンでは、既存のテーブル定義があり、ファイル内で同じバイトを共有する 2 つの列定義が含まれています。これは重複する列定義を作成することになるので修正が必要です。

目的

このレッスンの目的は、DDF Builder でファイルを開き、DDF Builder によってこの定義になんらかの変更が行われるかどうかを確認することです。DDF Builder が実施する変更を調べ、重複している列定義を修正するために必要となる変更について検討します。

メモ DDF Builder はファイル内の重複する列定義を修正するための解決策を提示し、それらの変更を保存できるようにします。

必要な知識

このレッスンでは、OVERLAP.MKD というファイルを使用します。このファイルは Tutorial2 フォルダー内にあります。デフォルトのインストール先にインストールしている場合、このフォルダーは次の場所にあります。

<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2

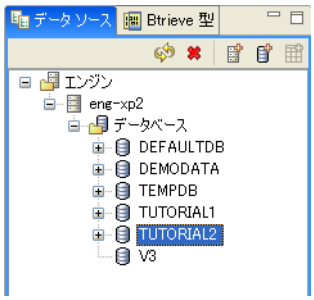
このフォルダーは Tutorial2 データベースの一部です。

メモ このチュートリアルのデータにアクセスするためには、このデータベースを指し示す DSN（データ ソース名）が必要です。まだ DSN を作成していない場合は、「データ ソース名 (DSN) の作成」を参照してください。

Btrieve ファイルを開く

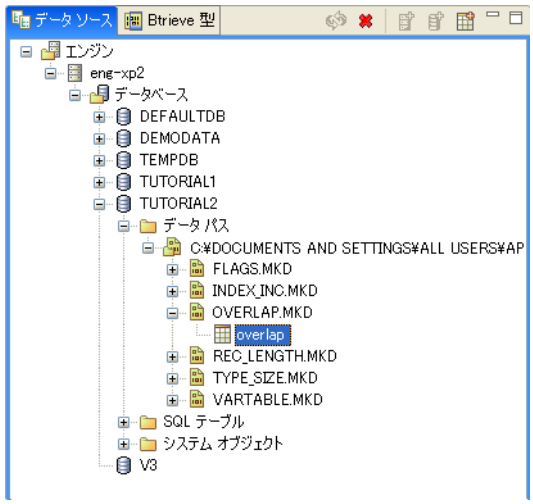
すぐ前のレッスンから引き続き作業していれば DDF Builder は起動しているはずです。終了していた場合は、DDF Builder を起動してください。

1 DDF Builder のデータ ソース エクスプローラーで、[データベース] ノードを展開し、一覧から Tutorial2 データベースを見つけます。



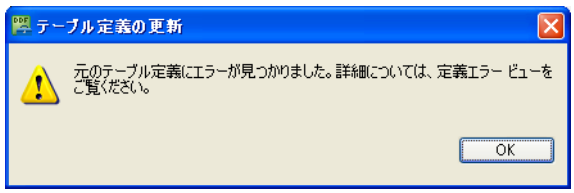
2 TUTORIAL2 データベース アイコンをダブルクリックしてそのノードを展開します。

3 ノードをさらに展開して OVERLAP.MKD ファイルを探し、それに関連付けられている SQL テーブルを表示します。



4　overlap SQL テーブル名をダブルクリックするか、右クリックして［**テーブル定義の編集**］をクリックします。

テーブル定義エディターでは、次のメッセージが表示されます。



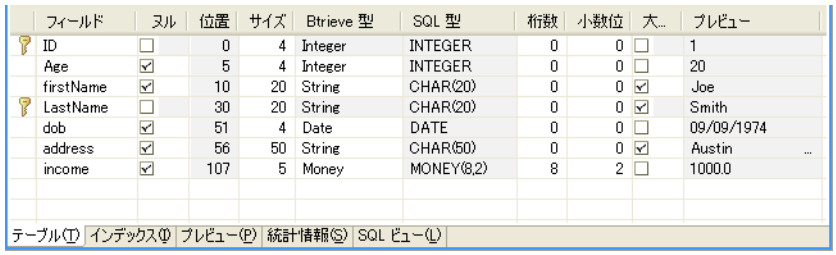
このメッセージは、DDF Builder で既存のテーブル定義を分析したら、その定義に問題があったということを知らせています。この結果、DDF Builder で既存のテーブル定義を開いて表示するために、いくつか修正を行う必要がありました。

5　［OK］をクリックすると、このメッセージがクリアされテーブル定義が表示されます。

ヒント　発生する可能性がある定義エラーの詳細については、「テーブル定義エラー」を参照してください。

不一致を探す

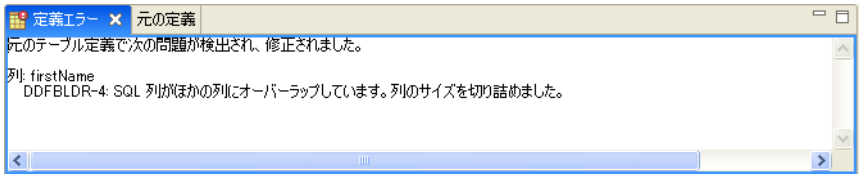
まずは DDF Builder インターフェイスで、元の定義と DDF Builder による変更後の定義との違いを確認し、報告されたエラーについても再調査します。



テーブル定義エディターのグリッド データ ビューでは、既存のテーブル定義に対して DDF Builder が行った変更を反映したテーブル定義が表示されます。

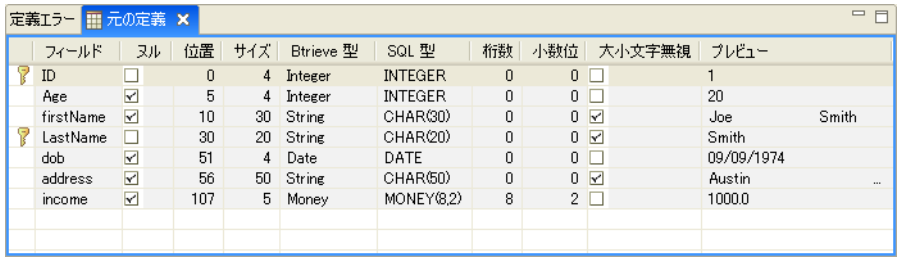
変更または追加されたフィールドであること示す不明アイコンやその他の視覚的なインジケーターはありません。

定義エラー ビューでは DDF Builder によって検出および変更された問題を表示します。



DDF Builder によって変更が行われる前の元のテーブル定義は、元の定義ビューでいつでも見ることができます。

メモ DDF Builder が行った変更は自動的に保存されません。DDF Builder による修正は保存する必要があります。



元の定義と変更後の定義はすぐに比較できますが、最初は DDF Builder によって報告されるエラーを見てください。

エラーについて理解する

定義エラー ビューでは以下のことを通知します。

- 問題が発生した場所
- DDF Builder で検出された問題
- DDF Builder で元のテーブル定義に対して行った変更事項

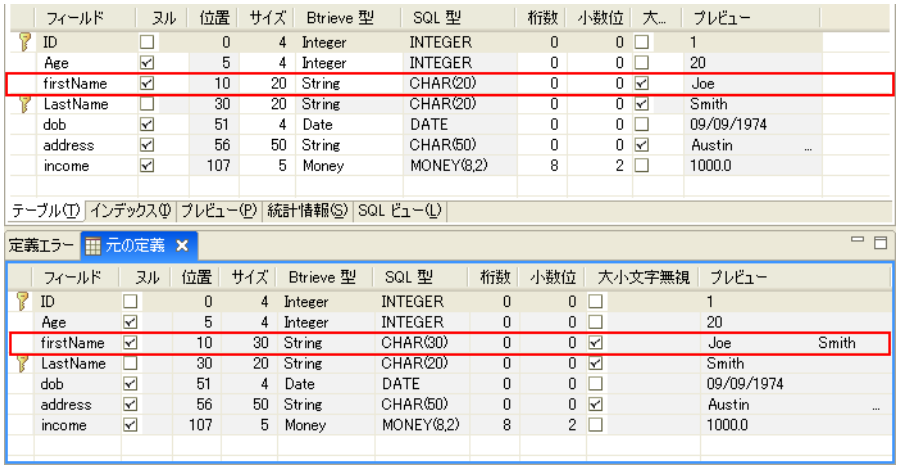
定義エラー ビューでは次の 1 つの問題が挙げられています。この問題について詳しく見てみましょう。

問題が発生した場所	DDF Builder で検出された問題	必要な対処
列：firstName	DDFBLDR-4：SQL 列がほかの列にオーバーラップしています。列のサイズを切り詰めました。	DDF Builder が指名した列（firstName）はほかの列と重なっています。重複する列をなくし、テーブル定義と Btrieve ファイルが同じレコード長になるようにするために、DDF Builder は列のサイズを短くしました。

DDF Builder によって報告されたエラーは列定義の重複です、これは、テーブル定義に同じバイトを共有する 2 つの列の定義が含まれていることを意味します。

DDF Builder では、firstName 列が次の列と重ならないようにするために、この列のサイズを自動的に削減します。これで列の長さが一致し、複数の列にわたって定義されているバイトがなくなります。

グリッド データ ビューと元の定義を見比べると、重複している列を確認することができます。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大...	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
LastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

定義エラー 元の定義

フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	30	String	CHAR(30)	0	0	☑	Joe Smith
LastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

変更後の定義で firstName フィールドのサイズが 20 であるのに対し、元の定義ではサイズが 30 になっていることに注目してください。元の定義では、[プレビュー] 列にファースト ネームの "Joe" とラスト ネームの "Smith" の両方が表示されています。変更後の定義ではこのフィールドのサイズが小さくなり、[プレビュー] 列にはファースト ネームのみが適切に表示されています。

変更を受け入れるまたは拒否する

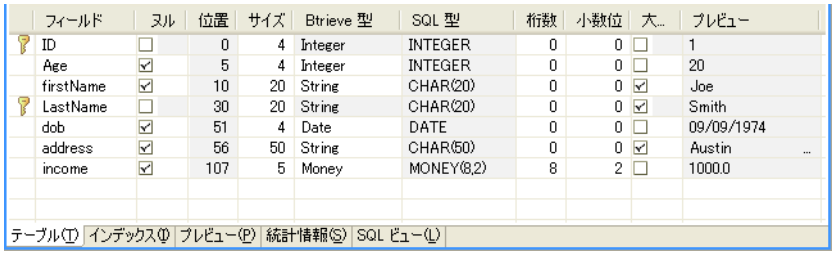
元の定義と DDF Builder によって変更された定義を比較し、それに加えてデータがどのように書式設定されたかを検証することは、DDF Builder によって行われた変更を受け付けるかどうかを判断するのに役立ちます。

DDF Builder が行った変更の結果、データが期待どおりの書式で表示され、列の長さも一致することが確認できます。

両ファイルの違いを調べ、DDF Builder による変更を理解し、データが適切であることを検証できたら、いつでもその変更を受け入れることができます。

テーブル定義を保存する

テーブル定義の変更を行ったのが、ユーザーであろうと DDF Builder であろうと、その変更を有効にするためにはそれを保存する必要があります。テーブル定義を保存する前に、もう一度テーブル定義と [プレビュー] タブを見て、すべてのフィールドが定義済みで各バイトが明確にされていることを確認すると良いでしょう。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大...	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
LastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

メモ テーブル定義で Btrieve 型が "Unknown"（不明）であったり、明確に割り当てられていないバイトが含まれているフィールドがあると、そのテーブル定義は保存することができません。

すべての Btrieve 型が定義済みで、レコード内のすべてのバイトが明確に割り当てられていることがわかります。これでテーブル定義を保存できます。

1　メニューから、[**ファイル**] > [**保管**] を選択するとテーブル定義が保存されます。

これで、列定義の重複を修正するために DDF Builder が変更したテーブル定義は、現在の定義として保存できました。

終わりに

このレッスンでは、列定義の重複が DDF Builder でどのように処理されるか、また列の重複を解決するために DDF Builder が実施する変更についてご紹介しました。DDF Builder によって変更された定義と元の定義を比較して、その変更を受け入れるか、あるいは拒否するかを判断する方法も示しました。

レッスン 5 ー ファイル / フィールド フラグの不一致

シナリオ

このレッスンでは、テーブル定義で設定されている SQL 列フラグと一致しないフラグ設定のフィールドを持つ Btrieve ファイルがあります。

目的

このレッスンの目的は、DDF Builder でファイルを開き、DDF Builder によってこの定義になんらかの変更が行われるかどうかを確認することです。DDF Builder が実施する変更を調べ、フラグの不一致を修正するために必要な変更があればその変更について検討します。

必要な知識

このレッスンでは、FLAGS.MKD というファイルを使用します。このファイルは Tutorial2 フォルダー内にあります。デフォルトのインストール先にインストールしている場合、このフォルダーは次の場所にあります。

<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2

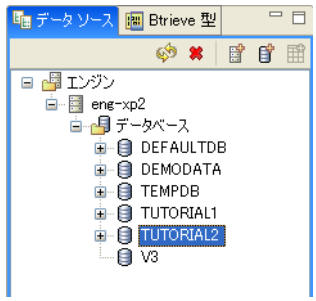
このフォルダーは Tutorial2 データベースの一部です。

メモ このチュートリアルのデータにアクセスするためには、このデータベースを指し示す DSN（データ ソース名）が必要です。まだ DSN を作成していない場合は、「データ ソース名 (DSN) の作成」を参照してください。

Btrieve ファイルを開く

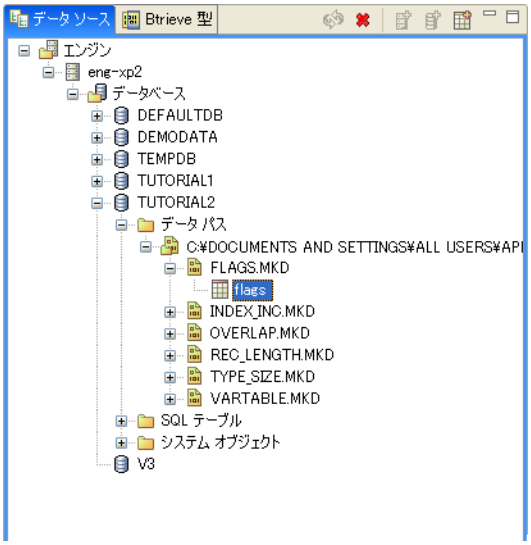
すぐ前のレッスンから引き続き作業していれば DDF Builder は起動しているはずです。終了していた場合は、DDF Builder を起動してください。

1 DDF Builder のデータ ソース エクスプローラーで、［データベース］ノードを展開し、一覧から Tutorial2 データベースを見つけます。



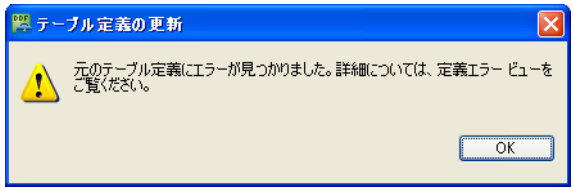
2 TUTORIAL2 データベース アイコンをダブルクリックしてそのノードを展開します。

3 ノードをさらに展開して FLAGS.MKD ファイルを探し、それに関連付けられている SQL テーブルを表示します。



4　flags SQL テーブル名をダブルクリックするか、右クリックして［**テーブル定義の編集**］をクリックします。

テーブル定義エディターでは、次のメッセージが表示されます。



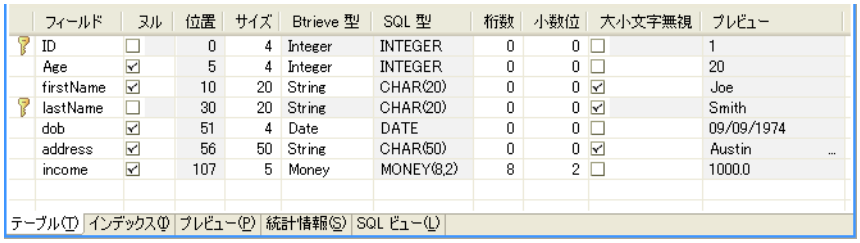
このメッセージは、DDF Builder で既存のテーブル定義を分析したら、その定義に問題があったということを知らせています。この結果、DDF Builder で既存のテーブル定義を開いて表示するために、いくつか修正を行う必要がありました。

5　［OK］をクリックすると、このメッセージがクリアされテーブル定義が表示されます。

ヒント　発生する可能性がある定義エラーの詳細については、「テーブル定義エラー」を参照してください。

不一致を探す

まずは DDF Builder インターフェイスで処理対象の不一致を確認します。まずは、グリッド データ ビューと定義エラー ビューを見てください。



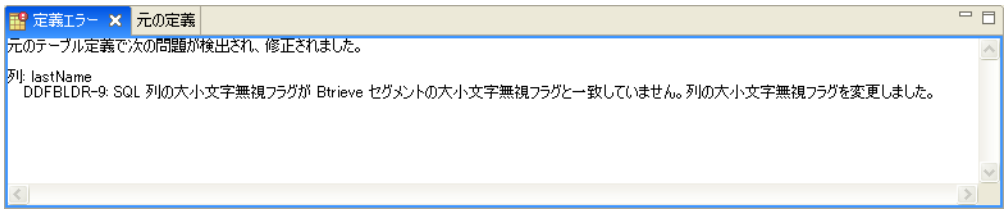
テーブル定義エディターのグリッド データ ビューでは、既存のテーブル定義に対して DDF Builder が行った変更を反映したテーブル定義が表示されています。

メモ DDF Builder が行った変更は自動的に保存されません。DDF Builder による修正は保存する必要があります。

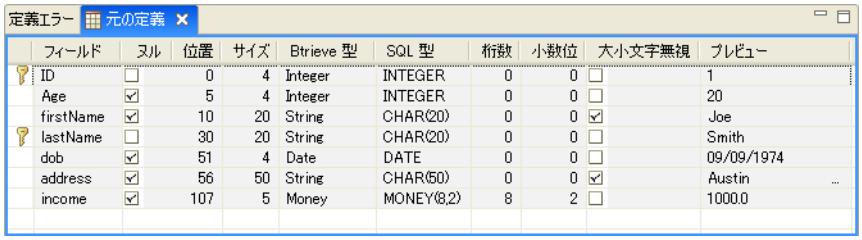
このレッスンの場合、DDF Builder は注目すべき列について視覚的なインジケーターを何も示しません。フィールドが変更されていることを示す不明な列インジケーターはありません。

ヒント グリッド データ ビューにおける属性の詳細については、「グリッド データ ビューでのフィールド属性」を参照してください。

定義エラー ビューでは DDF Builder によって検出および変更された問題を表示します。



ここで、元のテーブル定義を見てみましょう。DDF Builder によって変更が行われる前の元のテーブル定義は、元の定義ビューで見ることができます。



テーブル定義の不一致の詳しい説明に進む前に、DDF Builder によって報告された定義エラーを理解しておいてください。

エラーについて理解する

定義エラー ビューでは以下のことを通知します。

- 問題が発生した場所
- DDF Builder で検出された問題
- DDF Builder で元のテーブル定義に対して行った変更事項

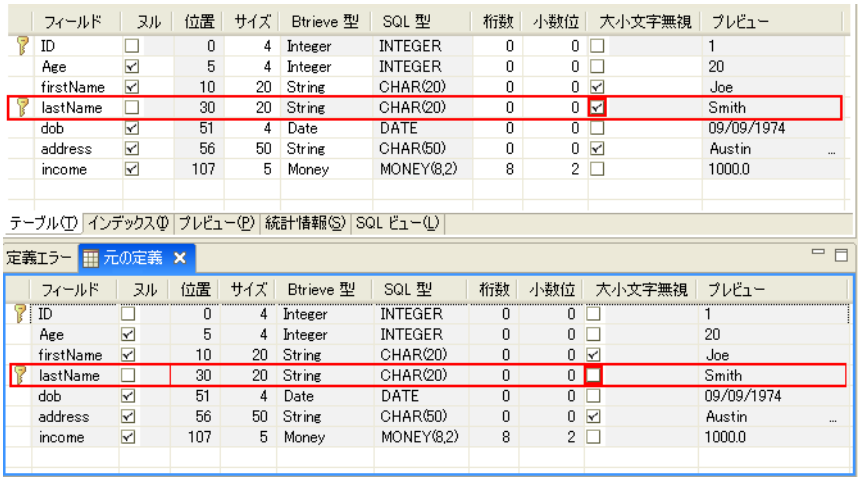
定義エラーでは次の 1 つの問題を挙げています。

問題が発生した場所	DDF Builder で検出された問題	必要な対処
列：lastName	DDFBLDR-9：SQL 列の大小文字無視フラグが Btrieve セグメントの大小文字無視フラグと一致していません。列の大小文字無視フラグが変更されます。	エラー対象のフィールドに大小文字無視フラグを設定するかどうかを検証し、DDF Builder による変更を受け入れるか、または拒否する必要があります。

このエラーは、SQL 列に対して大小文字無視フラグが設定されているが、対応する Btreive セグメントにはこのフラグが設定されていないことを示しています。DDF Builder は Btrieve ファイルを変更しないので、SQL テーブ

ルは Btrieve ファイルの仕様に一致するよう変更されます。この状況を確かめるには、現在のテーブル定義を元の定義と比較します。

定義エラーでは lastName 列の大小文字無視の設定に問題があることを示しています。現在のテーブル定義と元のテーブル定義の lastName フィールドを比較すると、この違いがすぐにわかります。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
lastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

定義エラー 元の定義

フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
lastName	☐	30	20	String	CHAR(20)	0	0	☐	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

この比較によって、以下のことが判断できます。

- 元の定義では、lastName フィールドに対して大小文字無視フラグが**設定されていませんでした**。
- 現在の（DDF Builder によって更新された）定義では、lastName フィールドに対して大小文字無視フラグが**設定されています**。

DDF Builder は Btrieve ファイルに基づいて変更を行います。DDF Builder では Btrieve ファイルを変更できません。変更できるのは Btrieve ファイルの構造を決定するテーブル定義のみであることを覚えておいてください。そのような訳で、DDF Builder は（大小文字無視フラグのない）元の定義を変更して、lastName フィールドで大小文字無視フラグを利用できるようにしました。

メモ lastName フィールドはインデックスとして使用されます。このため、lastName フィールドの**大小文字無視**フラグの設定をクリアすることはできません。

変更を受け入れるまたは拒否する

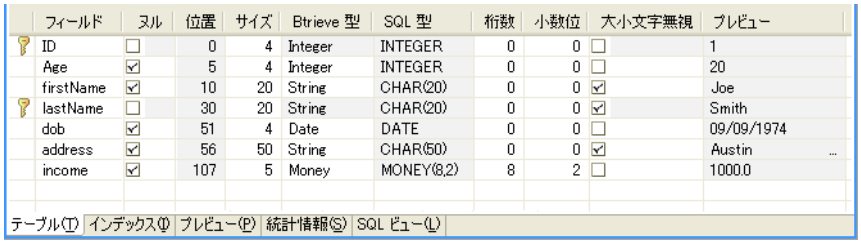
定義エラーの原因を究明し、DDF Builder が行った変更について理解しました。ここで、DDF Builder がテーブル定義に対して行った変更を受け入れるか、または拒否する必要があります。

DDF Builder が行った変更が正しくなければ、その変更を拒否して最初からテーブル定義を作成し直してください。これは、インデックスとして指定されているフィールドの大小文字無視の設定は変更できないためです。

ただし、このレッスンの目的は DDF Builder が行った変更を受け入れることです。lastName フィールドには大小文字無視フラグを設定します。

テーブル定義を保存する

テーブル定義エラーと変更について確認できたら、この作業を保存して変更を有効にしてください。作業を保存する前に、もう一度テーブル定義を見直してください。データ グリッド ビューは次のようになります。



	フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
🔑	ID	☐	0	4	Integer	INTEGER	0	0	☐	1
	Age	☑	5	4	Integer	INTEGER	0	0	☐	20
	firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
🔑	lastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
	dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
	address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
	income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

Btrieve 型がすべて定義されていること、未定義のフィールドが残っていないこと、またすべてのバイトが明確に割り当てられていることが一目でわかります。これでテーブル定義を保存できます。

1　メニューから、［**ファイル**］>［**保管**］を選択するとテーブル定義が保存されます。

更新されたテーブル定義が保存され、lastName フィールドに対して大小文字無視フラグを設定できました。

終わりに

このレッスンでは、Btrieve ファイルと既存のテーブル定義でフラグ設定が異なる場合に DDF Builder でどのように処理されるかをご紹介しました。DDF Builder によって行われた変更を、グリッド データ ビューと元の定義ビューを使用して比較できることも示しました。また、DDF Builder ではインデックスとして設定されているフィールドに対する変更が行えないことも示しました。DDF Builder によって行われた変更を拒否する、または受け入れるための方針についても説明しました。

レッスン 6 – インデックスの不一致

シナリオ

このレッスンでは、テーブル定義のインデックス設定と一致しないインデックス設定を持つ Btrieve ファイルがあります。

目的

このレッスンの目的は、DDF Builder でファイルを開き、DDF Builder によってこの定義になんらかの変更が行われるかどうかを確認することです。DDF Builder が実施する変更を調べ、インデックスの不一致を修正するために必要な変更について検討します。

ヒント DDF Builder はインデックスの不一致を修正するための解決法を提示し、それらの変更を保存できるようにします。

必要な知識

このレッスンでは、INDEX_INC.MKD というファイルを使用します。このファイルは Tutorial2 フォルダー内にあります。デフォルトのインストール先にインストールしている場合、このフォルダーは次の場所にあります。

<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2

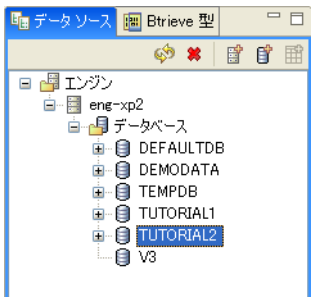
このフォルダーは Tutorial2 データベースの一部です。

メモ このチュートリアルのデータにアクセスするためには、このデータベースを指し示す DSN（データ ソース名）が必要です。まだ DSN を作成していない場合は、「データ ソース名（DSN）の作成」を参照してください。

Btrieve ファイルを開く

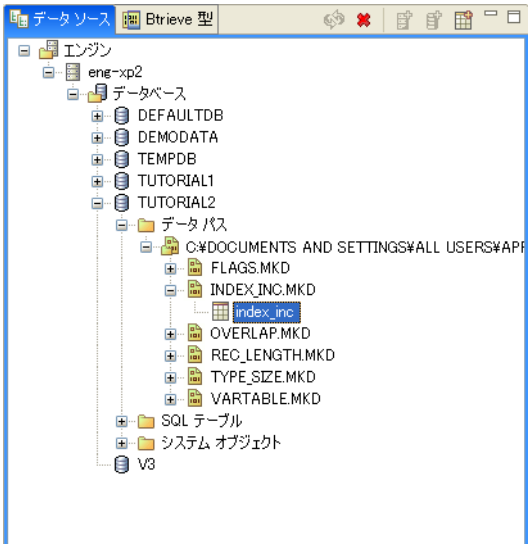
すぐ前のレッスンから引き続き作業していれば DDF Builder は起動しているはずです。終了していた場合は、DDF Builder を起動してください。

1 DDF Builder のデータ ソース エクスプローラーで、［データベース］ノードを展開し、一覧から Tutorial2 データベースを見つけます。



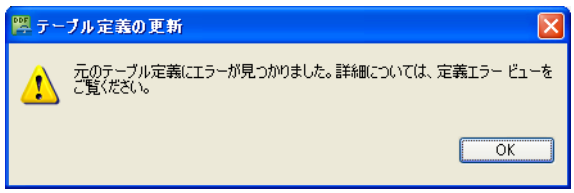
2 TUTORIAL2 データベース アイコンをダブルクリックしてそのノードを展開します。

3 ノードをさらに展開して INDEX_INC.MKD ファイルを探し、それに関連付けられている SQL テーブルを表示します。



4 index_inc SQL テーブル名をダブルクリックするか、右クリックして［**テーブル定義の編集**］をクリックします。

テーブル定義エディターでは、次のメッセージが表示されます。



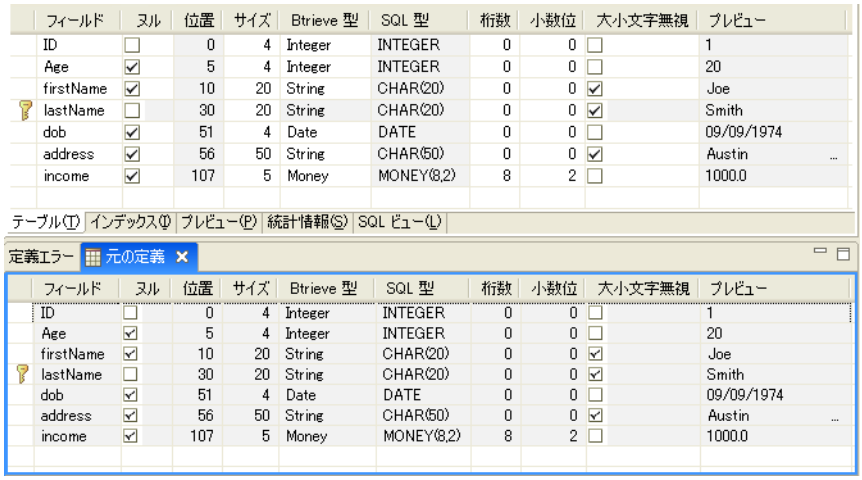
このメッセージは、DDF Builder で既存のテーブル定義を分析したら、その定義に問題があったということを知らせています。この結果、DDF Builder で既存のテーブル定義を開いて表示するために、いくつか修正を行う必要がありました。

5 ［OK］をクリックすると、このメッセージがクリアされテーブル定義が表示されます。

ヒント 発生する可能性がある定義エラーの詳細については、「テーブル定義エラー」を参照してください。

不一致を探す

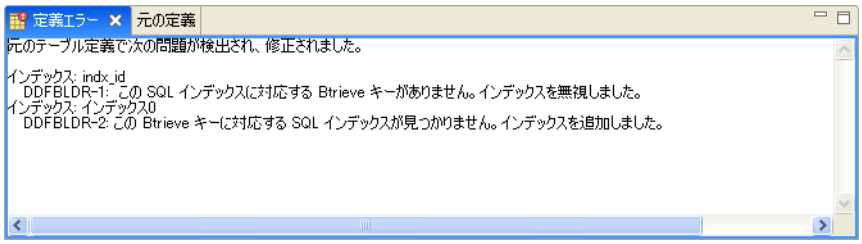
まずは DDF Builder インターフェイスで検出された不一致を確認します。これは、グリッド データ ビューと元の定義を比較すればすぐにわかります。



このレッスンの場合、DDF Builder は注目すべき列について視覚的なインジケーターを何も示しません。フィールドが変更されていることを示す不明な列インジケーターはありません。

大変興味深いことに、これらの 2 つのテーブル定義を比較しても違いが見つかりません。DDF Builder では変更を行っていますが、DDF Builder が修正した問題はインデックスに関するものなので、テーブル定義上に表れません。

インデックス情報についてより詳しく見る前に、定義エラー ビューに挙げられている問題を調査しましょう。定義エラー ビューでは DDF Builder によって検出および変更された問題を表示します。



メモ DDF Builder が行った変更は自動的に保存されません。DDF Builder による修正は保存する必要があります。

エラーについて理解する

定義エラー ビューでは以下のことを通知します。

- 問題が発生した場所
- DDF Builder で検出された問題
- DDF Builder で元のテーブル定義に対して行った変更事項

ヒント DDF Builder では Btrieve ファイルを変更できません。変更は Btrieve ファイルではなくテーブル定義に対して行われます。

定義エラーでは次の 2 つの問題を挙げています。

問題が発生した場所	DDF Builder で検出された問題	必要な対処
インデックス : indx_id	DDFBLDR-1 : この SQL インデックスに対応する Btrieve キーがありません。インデックスを無視しました。	テーブル定義で見つかった SQL インデックスに対応するキーが Btrieve ファイルに存在しません。その結果、DDF Builder ではこのインデックスが無視されるため、実行できる操作は DDF Builder による変更の確認のみです。
インデックス : インデックス 0	DDFBLDR-2 : この Btrieve キーに対応する SQL インデックスが見つかりません。インデックスを追加しました。	Btrieve ファイルにキーがあるため、それに対応するインデックスが DDF Builder によってテーブル定義に追加されました。テーブル定義のインデックスを検証して、DDF Builder が行った変更を受け入れる必要があります。

1 番目のエラーは、Btrieve ファイルに存在しないインデックスが既存のテーブル定義に含まれていることを示します。DDF Builder はテーブル定義を変更するだけで Btrieve ファイル自体を変更することはできないため、このインデックスは無視されます。これは、存在しないインデックスを使用しないようにするためにも知っておくことが重要です。

2 番目のエラーはこの逆の状況です。既存のテーブル定義に、Btrieve ファイルで定義されているインデックスがありません。DDF Builder は Btrieve ファイルのインデックスを表すために、これまで定義されていなかった SQL インデックスを追加します。

DDF Builder が作成したインデックスに名前を割り当て、そのテーブル定義のインデックスに対する変更を受け入れます。

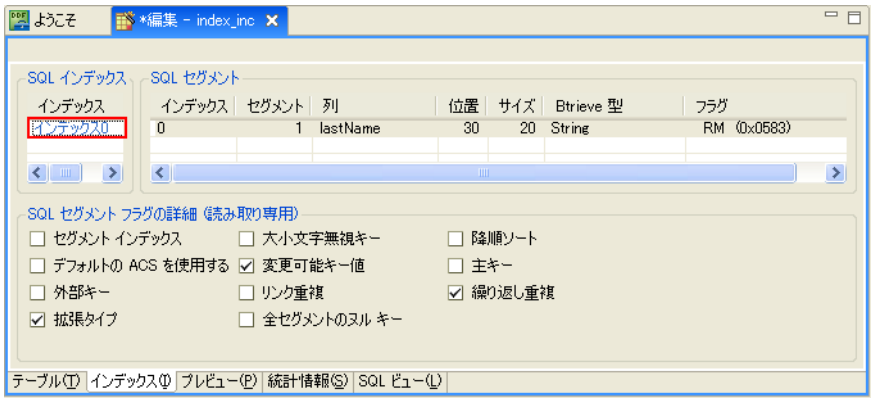
メモ DDF Builder によって作成されたインデックスがあり、それがもう必要でない場合は、ファイルおよびテーブル定義からそのインデックスを削除することができます。同様に、DDF Builder がテーブル定義を変更したときに、これまで定義されていなかったインデックスを追加する必要があることに気付くことができます。どちらの場合でも、『*Zen Programmer's Guide*』でインデックスの追加と削除に関する説明を参照してください。

インデックスに名前を付ける

1 テーブル定義エディターで［**インデックス**］タブを選択します。

2 ［インデックス］列の " インデックス 0" エントリをダブルクリックします。

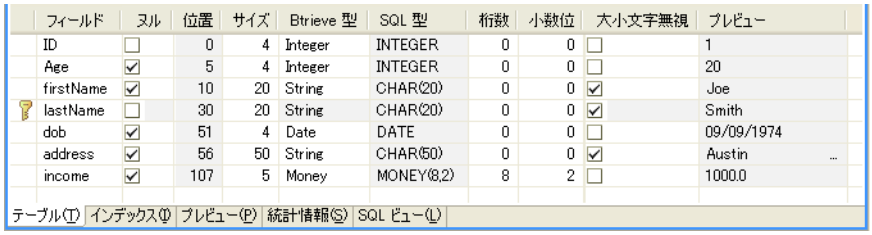
3 インデックスの名前として「index_lname」と入力します。



メモ Btrieve ファイルに定義されたインデックスを見るには、［統計情報］タブをクリックします。

テーブル定義を保存する

DDF Builder によってテーブル定義に追加されたインデックスを検証したので、この作業を保存して変更を有効にしてください。作業を保存する前に、もう一度テーブル定義を見直してください。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
lastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
address	☑	56	50	String	CHAR(50)	0	0	☑	Austin
income	☑	107	5	Money	MONEY(8,2)	8	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

Btrieve 型はすべて定義され、不明なフィールドも無く、またすべてのバイトが明確に割り当てられています。これでテーブル定義を保存できます。

1　メニューから、［**ファイル**］>［**保管**］を選択するとテーブル定義が保存されます。

終わりに

おめでとうございます。これでテーブル定義の保存が成功しました。このレッスンでは、DDF Builder が認識したインデックスの不一致についてご紹介しました。また、DDF Builder が行った変更と更新されたテーブル定義の保存方法についても説明しました。

レッスン 7 – 可変長レコードの不一致

シナリオ

このレッスンでは、テーブル定義で設定されている可変長レコードとは異なる可変長レコードを持つ Btrieve ファイルがあります。

目的

このレッスンの目的は、DDF Builder でファイルを開き、DDF Builder によってこの定義になんらかの変更が行われるかどうかを確認することです。DDF Builder が実施する変更を調べ、可変長レコードの不一致を修正するために必要となる変更について検討します。

メモ 現在、DDF Builder は可変長レコードの不一致を修正するための解決策を提示していません。

必要な知識

このレッスンでは、VARTABLE.MKD というファイルを使用します。このファイルは Tutorial2 フォルダー内にあります。デフォルトのインストール先にインストールしている場合、このフォルダーは次の場所にあります。

<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2

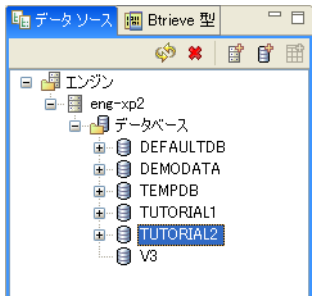
このフォルダーは Tutorial2 データベースの一部です。

メモ このチュートリアルのデータにアクセスするためには、このデータベースを指し示す DSN（データ ソース名）が必要です。まだ DSN を作成していない場合は、「データ ソース名（DSN）の作成」を参照してください。

Btrieve ファイルを開く

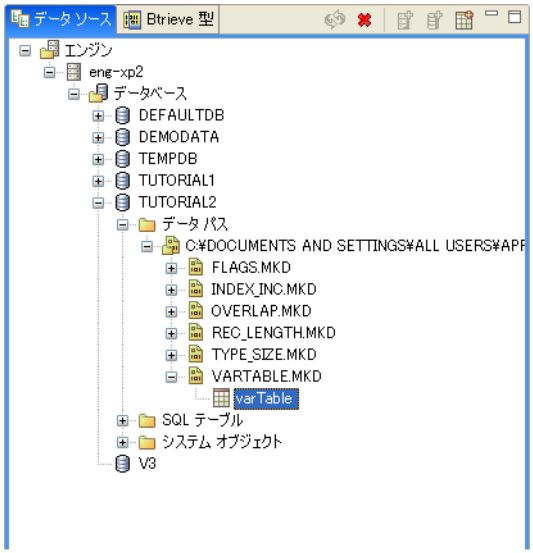
すぐ前のレッスンから引き続き作業していれば DDF Builder は起動しているはずです。終了していた場合は、DDF Builder を起動してください。

1 DDF Builder のデータ ソース エクスプローラーで、［データベース］ノードを展開し、一覧から Tutorial2 データベースを見つけます。



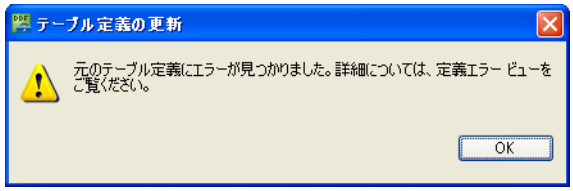
2 TUTORIAL2 データベース アイコンをダブルクリックしてそのノードを展開します。

3 ノードをさらに展開して VARTABLE.MKD ファイルを探し、それに関連付けられている SQL テーブルを表示します。



4　varTable SQL テーブル名をダブルクリックするか、右クリックして[**テーブル定義の編集**]をクリックします。

テーブル定義エディターでは、次のメッセージが表示されます。



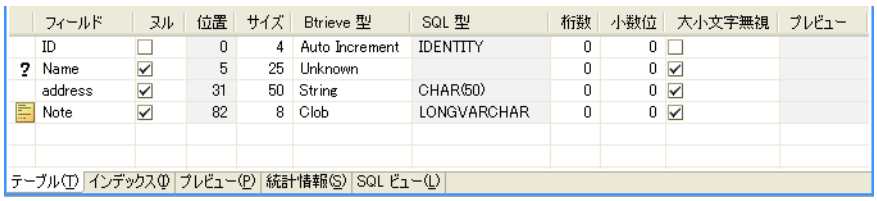
このメッセージは、DDF Builder で既存のテーブル定義を分析したら、その定義に問題があったということを知らせています。この結果、DDF Builder で既存のテーブル定義を開いて表示するために、いくつか修正を行う必要がありました。

5　［OK］をクリックすると、このメッセージがクリアされテーブル定義が表示されます。

ヒント　発生する可能性がある定義エラーの詳細については、「テーブル定義エラー」を参照してください。

不一致を探す

まずは DDF Builder インターフェイスで、元の定義と DDF Builder による変更後の定義との違いを確認し、報告されたエラーについても再調査します。



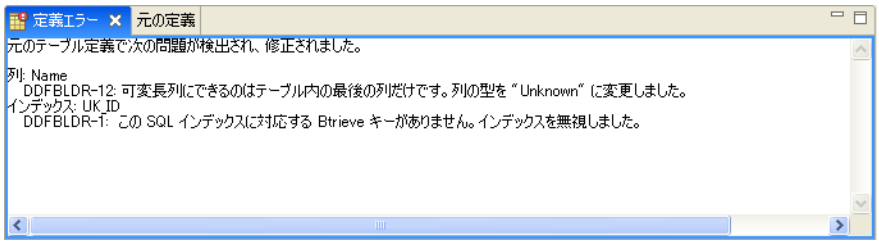
テーブル定義エディターのグリッド データ ビューでは、変更済みのテーブル定義が示されます。

メモ　DDF Builder が行った変更は自動的に保存されません。DDF Builder による修正は保存する必要があります。

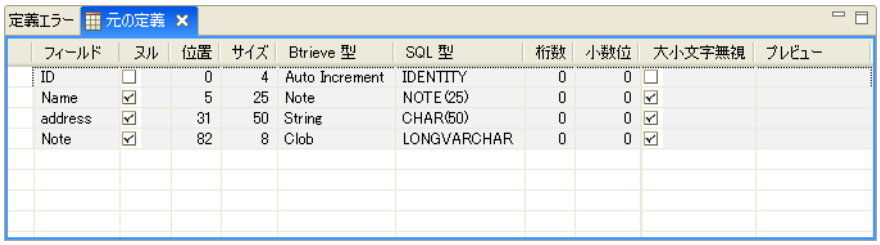
DDF Builder では、変更または追加されたフィールドに対して不明な列インジケーターを追加し、注目すべき列であることを視覚的に示します。また、可変長列インジケーターが可変長の列に追加されます。

ヒント グリッド データ ビューにおける属性の詳細については、「グリッド データ ビューでのフィールド属性」を参照してください。

定義エラー ビューでは DDF Builder によって検出および変更された問題を表示します。



DDF Builder によって変更が行われる前の元のテーブル定義は、元の定義ビューで見ることができます。



エラーについて理解する

定義エラー ビューでは以下のことを通知します。

- 問題が発生した場所
- DDF Builder で検出された問題
- DDF Builder で元のテーブル定義に対して行った変更事項

定義エラーでは次の 2 つの問題を挙げています。

問題が発生した場所	DDF Builder で検出された問題	必要な対処
列：Name	DDFBLDR-12：可変長列にできるのはテーブル内の最後の列だけです。列の型を "Unknown" に変更しました。	既存の SQL テーブル定義にはレコードの途中に NOTE または LVAR データ型があります。DDF Builder では、可変長列はレコード内の最後に配置する必要があると判断し、このデータ型を "Unknown"（不明）に変更します。DDF Builder が "Unknown"（不明）に変更した列を定義する必要があります。
インデックス：UK_ID	DDFBLDR-1：この SQL インデックスに対応する Btrieve キーがありません。インデックスを無視しました。	テーブル定義で見つかった SQL インデックスに対応するキーが Btrieve ファイルに存在しません。その結果、DDF Builder ではこのインデックスが無視されるため、実行できる操作は DDF Builder による変更の確認のみです。

DDFBLDR-12 エラーは、テーブル定義の最後の列ではない可変長フィールドが原因で発生しています。このファイルにはレコードの最後の列以外の位置に可変長フィールドがあるため、DDF Builder ではこのフィールドのデータ型を "Unknown"（不明）に変更します。

ヒント 「レッスン 3 – 不正なデータ型とサイズ」では、列のデータ型がその列のサイズに対して不正だった場合の具体的な対処手順を提供しています。

可変長フィールドの問題を修正するには、不明なフィールドを適切なデータ型で定義します。これは次のセクションで説明します。

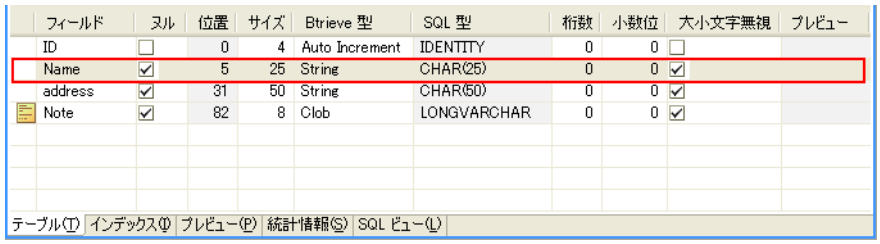
最後に報告されたエラーは、DDF Builder が既存のテーブル定義でインデックスを検出したが、それに対応するキーが Btrieve ファイルにはないという事象です。このレッスンの過程で、このエラーは解決されません。解決されるのは可変長フィールドについてだけです。インデックスに関するエラーを解決するためには、「レッスン 6 – インデックスの不一致」を参照してください。

不明なフィールドを定義する

テーブル定義エディターのグリッド データ ビューで、不明なフィールドを定義します。

1 Name フィールドをクリックして選択します。

2 ［Btrieve 型］リストから "String" を選択します。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Auto Increment	IDENTITY	0	0	☐	
Name	☑	5	25	String	CHAR(25)	0	0	☑	
address	☑	31	50	String	CHAR(50)	0	0	☑	
Note	☑	82	8	Clob	LONGVARCHAR	0	0	☑	

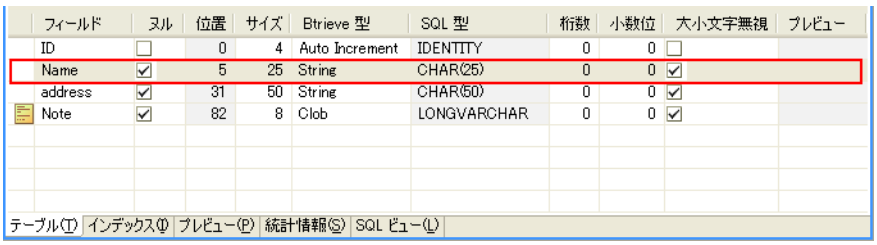
テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

メモ テーブル定義に BLOB または CLOB データ型が含まれている場合、テーブルの可変長部分にある列を編集することはお勧めしません。

テーブル定義を保存する

不明なフィールドを定義し、すべての問題を修正したら、テーブル定義を保存できます。

作業を保存する前に、もう一度テーブル定義を見直してください。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Auto Increment	IDENTITY	0	0	☐	
Name	☑	5	25	String	CHAR(25)	0	0	☑	
address	☑	31	50	String	CHAR(50)	0	0	☑	
Note	☑	82	8	Clob	LONGVARCHAR	0	0	☑	

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

テーブル定義に未定義のフィールドがなく、すべてのバイトが明確に割り当てられています。これでテーブル定義を保存できます。

1 メニューから、［**ファイル**］>［**保管**］を選択するとテーブル定義が保存されます。

終わりに

おめでとうございます。テーブル定義が保存され、可変長フィールドの修正に成功しました。このレッスンでは、レコードの最後にない可変長フィールドを DDF Builder でどのように処理するかをご紹介しました。また、フィールドを定義する方法と、ファイルに対して有効なテーブル定義を作成する方法についても説明しました。

レッスン 8 – レコード長の不一致

シナリオ

このレッスンでは、テーブル定義で設定されているレコード長とは異なるレコード長を持つ Btrieve ファイルがあります。

目的

このレッスンの目的は、DDF Builder でファイルを開き、DDF Builder によってこの定義になんらかの変更が行われるかどうかを確認することです。DDF Builder が実施する変更を調べ、レコード長の不一致を修正するために必要な変更について検討します。

ヒント DDF Builder はレコード長の不一致を修正するための解決策を提示し、それらの変更を保存できるようにします。

必要な知識

このレッスンでは、rec_length.mkd というファイルを使用します。このファイルは Tutorial2 フォルダー内にあります。デフォルトのインストール先にインストールしている場合、このフォルダーは次の場所にあります。

<*Application Data*>¥DDFBuilder¥tutorials¥tutorial2

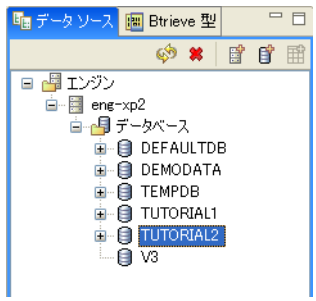
このフォルダーは Tutorial2 データベースの一部です。

メモ このチュートリアルのデータにアクセスするためには、このデータベースを指し示す DSN（データ ソース名）が必要です。まだ DSN を作成していない場合は、「データ ソース名（DSN）の作成」を参照してください。

Btrieve ファイルを開く

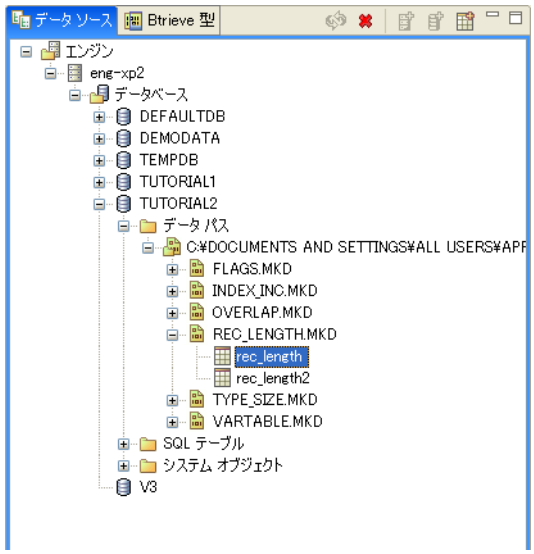
すぐ前のレッスンから引き続き作業していれば DDF Builder は起動しているはずです。終了していた場合は、DDF Builder を起動してください。

1 DDF Builder のデータ ソース エクスプローラーで、［データベース］ノードを展開し、一覧から Tutorial2 データベースを見つけます。



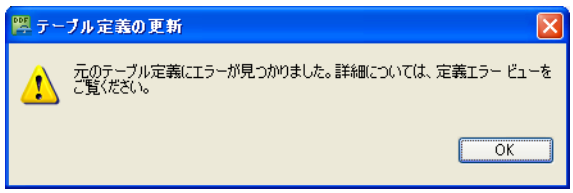
2 TUTORIAL2 データベース アイコンをダブルクリックしてそのノードを展開します。

3 ノードをさらに展開して REC_LENGTH.MKD ファイルを探し、それに関連付けられている SQL テーブルを表示します。



4 rec_length SQL テーブル名をダブルクリックするか、右クリックして［**テーブル定義の編集**］をクリックします。

テーブル定義エディターでは、次のメッセージが表示されます。



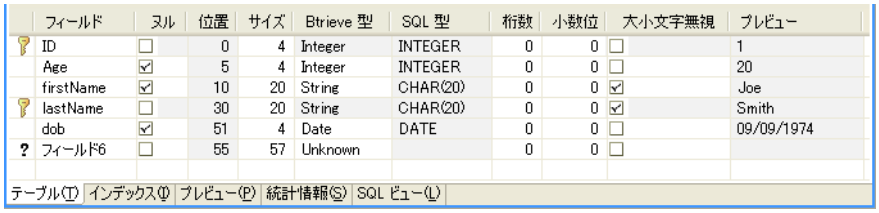
このメッセージは、DDF Builder で既存のテーブル定義を分析したら、その定義に問題があったということを知らせています。この結果、DDF Builder で既存のテーブル定義を開いて表示するために、いくつか修正を行う必要がありました。

5 ［OK］をクリックすると、このメッセージがクリアされテーブル定義が表示されます。

ヒント 発生する可能性がある定義エラーの詳細については、「テーブル定義エラー」を参照してください。

不一致を探す

まずは DDF Builder インターフェイスで、元の定義と DDF Builder による変更後の定義との違いを確認し、報告されたエラーについても再調査します。



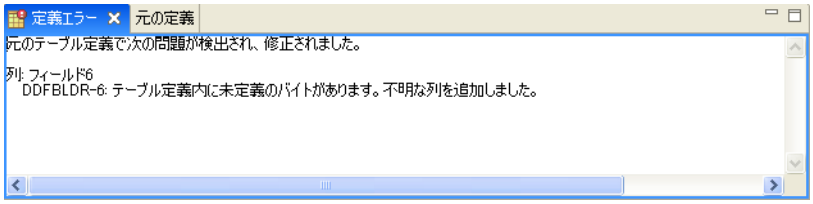
テーブル定義エディターのグリッド データ ビューでは、変更済みのテーブル定義が示されます。

メモ DDF Builder が行った変更は自動的に保存されません。DDF Builder による修正は保存する必要があります。

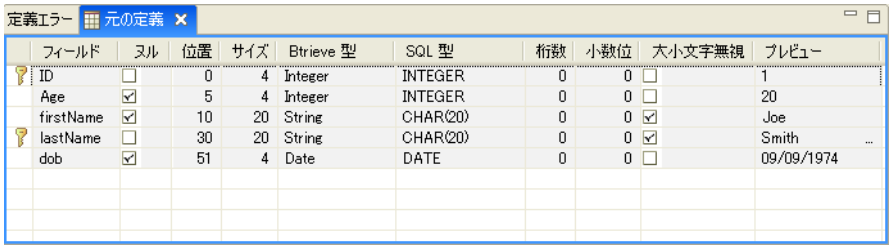
DDF Builder では、変更または追加されたフィールドに対して不明な列インジケーターを追加し、注目すべき列であることを視覚的に示します。

ヒント グリッド データ ビューにおける属性の詳細については、「グリッド データ ビューでのフィールド属性」を参照してください。

定義エラー ビューでは DDF Builder によって検出および変更された問題を表示します。



DDF Builder によって変更が行われる前の元のテーブル定義は、元の定義ビューで見ることができます。



エラーについて理解する

定義エラー ビューでは以下のことを通知します。

- 問題が発生した場所
- DDF Builder で検出された問題
- DDF Builder で元のテーブル定義に対して行った変更事項

定義エラーでは次の 1 つの問題を挙げています。

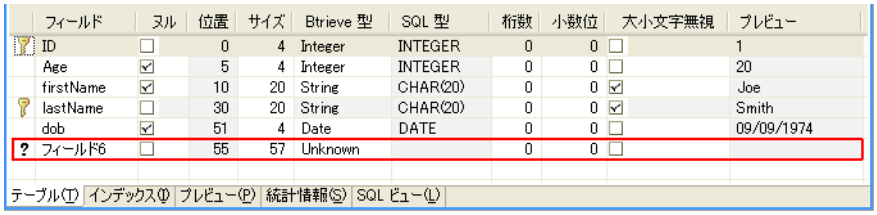
問題が発生した場所	DDF Builder で検出された問題	必要な対処
列：フィールド 6	DDFBLDR-6：テーブル定義内に未定義のバイトがあります。不明な列を追加しました。	DDF Builder が未定義のバイトを明らかにするために追加した不明な列を定義する必要があります。

DDF Builder によって報告されたエラーは、レコード長の不一致を示しています。

ヒント ［**統計情報**］タブをクリックすると、Btrieve ファイルの元のレコード長を調べることができます。

これは、レコードのすべてのバイトに当たるフィールドがテーブル定義に含まれていなかったことを意味します。このようなバイトを明らかにするために、DDF Builder は新しい列を不明なデータ型として作成しました。

すべてのバイトを定義して明確に割り当て、Btrieve ファイルと正確に一致するテーブル定義を作成します。グリッド データ ビューの下部にある " フィールド 6" の行を見てください。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
lastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
フィールド6	☐	55	57	Unknown		0	0	☐	

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

レコード長が Btrieve ファイルとテーブル定義で一致するよう、不明なフィールドを定義することができます。

フィールドを定義する

次に、テーブル定義エディターのグリッド データ ビューで、不明のフィールドを定義します。

1 グリッド データ ビューで最後の行をクリックして選択します。

2 " フィールド 6" フィールドを選択し、フィールド名として「City」と入力します。

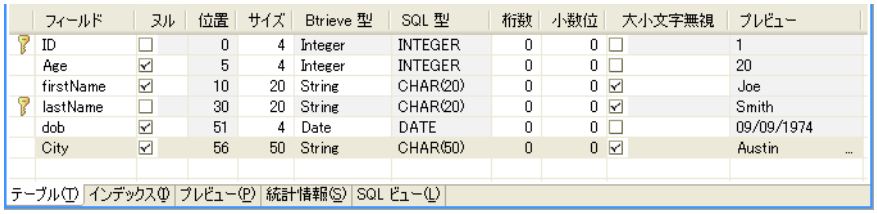
3 ［**ヌル**］チェック ボックスのチェックをオンにします。

4 ［**サイズ**］で値を 50 バイトに変更してください。

5 ［Btrieve 型］列のリストから "String" を選択します。

6 ［**大小文字無視**］チェック ボックスをオンにします。

テーブル定義は次のようになります。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
lastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
City	☑	56	50	String	CHAR(50)	0	0	☑	Austin

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

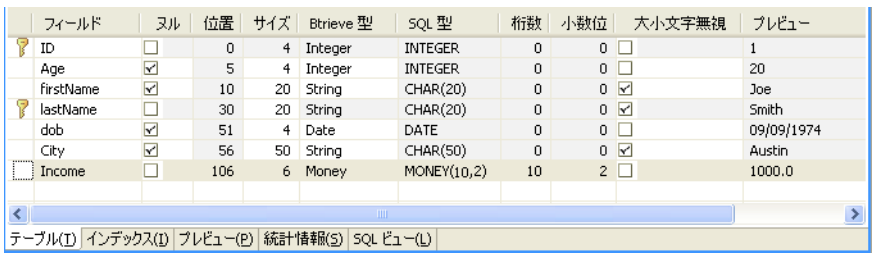
7 グリッド データ ビューで " フィールド 7" 行をクリックして選択します。

8 " フィールド 7" フィールドを選択し、フィールド名として「Income」と入力します。

9 ［Btrieve 型］列のリストから "Money" を選択します。

10 ［**桁数**］の値は 10 に、［**小数位**］の値は 2 にします。

テーブル定義は次のようになります。



フィールド	ヌル	位置	サイズ	Btrieve 型	SQL 型	桁数	小数位	大小文字無視	プレビュー
ID	☐	0	4	Integer	INTEGER	0	0	☐	1
Age	☑	5	4	Integer	INTEGER	0	0	☐	20
firstName	☑	10	20	String	CHAR(20)	0	0	☑	Joe
lastName	☐	30	20	String	CHAR(20)	0	0	☑	Smith
dob	☑	51	4	Date	DATE	0	0	☐	09/09/1974
City	☑	56	50	String	CHAR(50)	0	0	☑	Austin
Income	☐	106	6	Money	MONEY(10,2)	10	2	☐	1000.0

テーブル(T) インデックス(I) プレビュー(P) 統計情報(S) SQL ビュー(L)

これで不明なフィールドはすべて定義され、すべてのバイトは明確に割り当てられたので、テーブル定義を保存します。

定義を保存する

行った変更を保存して適用できるようテーブル定義を保存します。

1　メニューから、［**ファイル**］>［**保管**］を選択するとテーブル定義が保存されます。

終わりに

おめでとうございます。テーブル定義が保存され、レコード長の不一致が適切に修正されました。このレッスンでは、レコード長が異なる場合に DDF Builder でどのように処理されるかをご紹介しました。また、フィールドを定義する方法や、サイズを変更して列を分割したり、ファイルに対して有効なテーブル定義を作成する方法についても説明しました。