

TEKLYNX®

CODESOFT®

SETTING THE STANDARD



チュートリアル

チュートリアル

DOC-OEMCS2012-TU-JA-05/21/13

このガイドに含まれる情報は、契約の性質を有するものではなく、事前の予告なしに変更される可能性があります。

このガイドに記載されるソフトウェアは、ライセンス契約のもとで販売されます。ソフトウェアは、契約の条件に従う場合のみ、使用、コピーまたは複製することができます。

このガイドのいかなる部分も、**Teklynx Newco SAS** から書面での許可を受けずに、購入者の個人的な使用以外の目的で、いかなる形式や手段においても、複写、複製または送信することはできません。

©2013 **Teklynx Newco SAS**,

All rights reserved.

本マニュアルについて

表記に関する原則

本マニュアルは、以下の原則を用いて、さまざまなタイプの情報を区別しています。

- コマンドなど、インターフェース自体の用語は、**太字**で表記します。
- キーボードのキーは細字の大文字を四角で囲んで表記します。
例:「**SHIFT**キーを押してください。」
- 番号のついたリストは、実行する手順があることを示します。
- 接続詞「または」が段落の隣に表示された場合、特定のタスクを実行する別の手順があることを表します。
- メニューコマンドにサブメニューが含まれる場合、選択するメニュー名と、それに続くコマンドが**太字**で表記されます。したがって、「**ファイル > 開く**を選択」の場合、**ファイル**メニューを選び、**開く**コマンドを選択します。

製品について

このマニュアルは、特定の製品について記載されたものではなく、製品間で共通の概念について説明しています。従って、記載されている一部の機能は、製品によってはご利用いただけない場合があります。

購入されたソフトウェアで利用できる機能については、製品のオンラインヘルプやカタログをご覧ください。

目次

データベースへの接続	1
注意事項	1
ODBC	1
OLE DB	2
SQL クエリービルダー	5
起動	6
オブジェクトをクエリーに追加	8
オブジェクト・プロパティの編集	9
テーブルの結合	10
出力フィールドのソート	11
条件の定義	11
パラメーター化したクエリーの定義	12
クエリー結果グリッド	12
データベースマネージャー	15
データベース構造ウィンドウ	15
データベース修正ウィンドウ	19
データベースクエリーウィンドウ	22
印刷ウィンドウ	25
数式	27
数式データソース	27
関数について	27
演算子	28
数学関数	29
論理関数	31

文字関数	32
IF 関数に関する情報.....	39
実習: 特別なチェックデジットの作成	40

データベースへの接続

注意事項

本章では、ラベル(コンテナ)とデータベース(コンテンツ)の関連付けを行います。ODBC (Open DataBase Connectivity) または OLE DB 接続を使用して、これを行います。

データベースを使用すると、データを保管することができます。すべてのデータは、リレーションシップと呼ばれる、2次元のテーブルにまとめられます。テーブルのそれぞれの行はレコードと呼ばれます。レコードは、フィールド形式で、テーブルのさまざまな列に配置されるプロパティである、オブジェクトの管理を目的としています。

データベースには多くのテーブルが含まれます。特定のデータベース内で、さまざまなテーブルを関連付けるには、結合を使用します。本章の後で、結合を作成する方法について具体的な例をご覧ください。

ODBC

ODBC データソースを使用すると、幅広いデータベース管理システムに属するデータにアクセスすることができます。ODBC は、ラベルデザイン・ソフトウェアなどのアプリケーションと、特定の数のデータベースを簡単にリンクさせることができます。ソフトウェアには、いくつかの ODBC ドライバーが搭載されています。

これらのドライバーによって、最もよくあるタイプのデータベースにアクセスすることができます。

ドライバーの一部をご紹介します。

- Microsoft Access Driver (*.mdb)
- Microsoft Excel Driver (*.xls)
- Microsoft FoxPro Driver (*.dbf)

OLE DB

OLE DB は、種類、フォーマット、場所に関わらず、すべてのデータにアクセス可能とするインターフェースのセットです。それは、アクセス・インターフェース、クエリー・ドライバーなどのコンポーネントを提供します。これらのコンポーネントは、「プロバイダー」と呼ばれます。

ODBC データソースのインストールとデータのインポート

以下の例では、データベースがソフトウェアに接続されていない場合の接続プロセスを説明します。

ODBC データソースのインストール

以下に記載したプロセスでは、直接作成モードを使用します。必要に応じて、操作状況にあわせて表示されるメニューで Wizard(ウィザード)を選択し、これを使用することができます。

TKTraining.mdb データベースへの接続

1. CODESOFT で、**ツール > ODBC アドミニストレーター**を選択します。
2. **システム DSN タブ** をクリックして、**追加ボタン**をクリックします。


注意:システム・データソース名 (DSN)を用いて、データソースを定義することができます。これらのデータソースは、特定のコンピューター固有のものですが、特定のユーザーに限定されるわけではありません。必要な権利を持つすべてのユーザーが、システム DSN にアクセスできます。

3. Microsoft Access Driver を選択し、**完了ボタン**をクリックします。
4. データソース名ボックスに「TK Training CS Level 2」を入力します。

5. **選択** ボタンをクリックして、C:\Documents and Settings\All Users\WINDOWS\Documents\Teklynx\CODESOFT 9\Samples\Tutorial にある TKTraining.mdb データベースを選択します。
6. **オプション** ボタンをクリックします。読み取り専用オプションにチェックをつけます。このオプションを使用すると、読み取り/書き込みの問題を生じることなく、CODESOFT と同時にデータベースを開くことができます。
7. ODBC Microsoft Access セットアップのダイアログボックスで、OK ボタンをクリックします。

データのインポート

データベースがソフトウェアに接続されたら、それをドキュメントに接続しなければなりません。

1. PRODUCT_WS3 ラベルを開きます。
2. **データソース > データベース > クエリー**の作成と修正を選択します。
3. データソース選択リストで、TK Training CS Level 2 を選択します。
4. 選択テーブルリストで、「Fruits」を選択します。
データベース・フィールドが、選択フィールドリストに表示されます。
5. 「ProdName」、「Origin」、「Weight」および「Reference」フィールドを選択します。
6.  ボタンをクリックします。選択されたレコードを、アルファベットまたは数字、昇順あるいは降順にソートすることができます。
7. **ソートキー**として「Reference」、**ソート順**として「昇順」を選択します。
8. このクエリーを PRODUCT_WS4_ODBC.CSQ として保存します。
9. **OK** ボタンをクリックします。
変数が自動的に作成され、データソースビューのデータベースのブランチにリストされます。

オブジェクトがとることができる値を表示または印刷するには、ナビゲーションバーを使用します。クエリー結果ウィンドウから印刷することもできます。



変数オブジェクトの作成

1. データソースビューのデータベースのブランチにリストされた、作成された変数を選択し、ワークスペースにドラッグ & ドロップします。
2. 表示されるコンテキストメニューにおいて、テキストを選択します。


コマンド: データソース > テーブル参照 > 追加

1. データソース選択リストから、データソースを選択します。

注意: 新しいデータソースを作成するには、**新規作成** ボタンをクリックします。これにより、ウィザードを使用したり、ODBC データソースまたは OLEDB データソースを選択したりすることができます。

2. デフォルトでは、**標準作成モード** が有効となります。しかし、テーブル参照を実行するには、**高度な作成モード** である SQL を使用できます。

標準作成モード

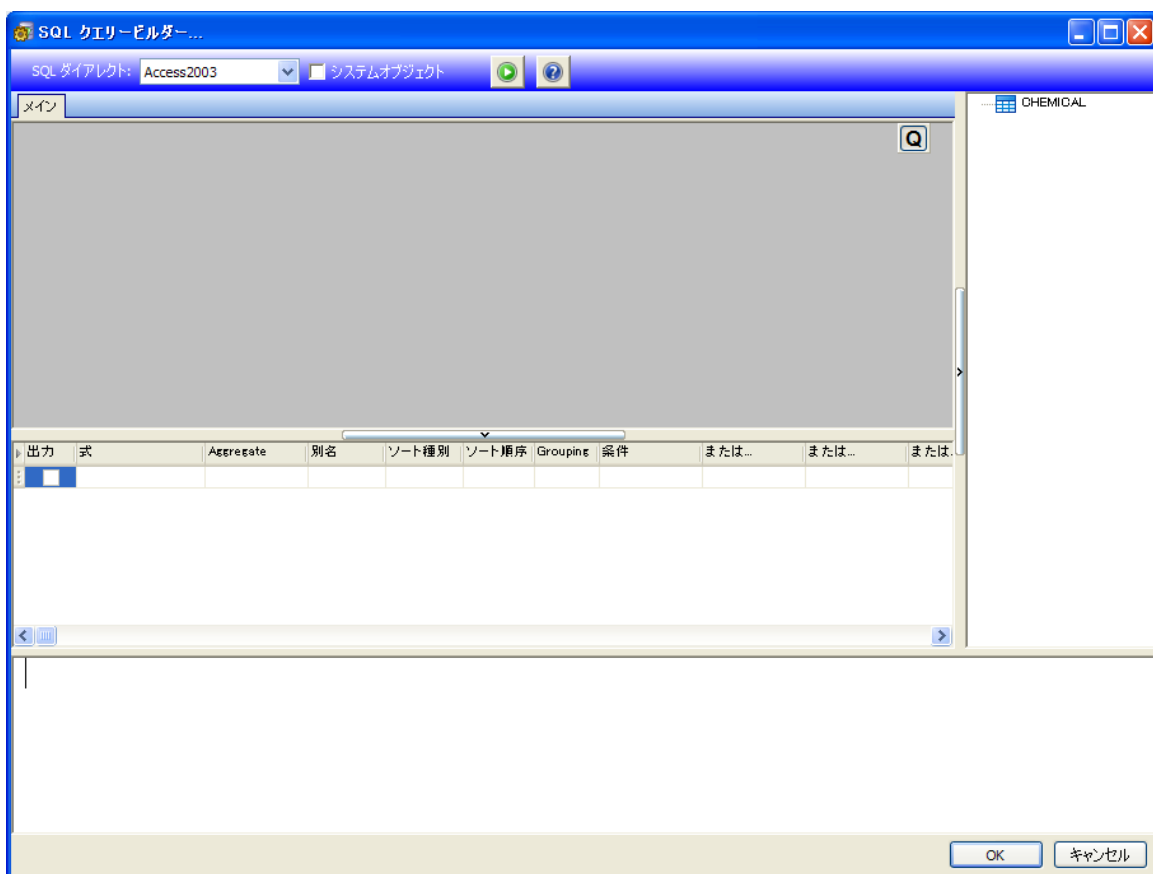
1. **選択テーブル** リストにおいて、検索を実行するテーブルを選択します。
2. **結果フィールド** リストにおいて、値を変数に変換するフィールドを選択します。
3.  ボタンをクリックして、行を追加します。
4. 検索を実行する外部テーブルのフィールドを選択します。
5. 検索値を含む、現在のドキュメントの変数を選択します。
6. **テスト** ボタンをクリックし、結果を表示します。

SQL 高度な作成モード

3. **SQL フォーマット作成モード**をクリックします。
4. SQL フォーマットでクエリーを入力します。

または

SQL クエリービルダー ボタンをクリックし、クエリービルダーにアクセスします。これにより、SQL データベース・クエリーを構築する際に、簡単に使用できるインターフェースをご利用いただけます。アプリケーションで、新しいリクエストを作成したり、既存のリクエストをグラフィカルに表示できます。



5. **テストボタン**をクリックして、**クエリー結果**ダイアログボックスに結果を表示します。

SQL クエリービルダー

クエリービルダーは、直感的で使いやすいビジュアルなクエリー構築インターフェースを介して、複雑な SQL クエリーを構築できるビジュアルなコンポーネントです。

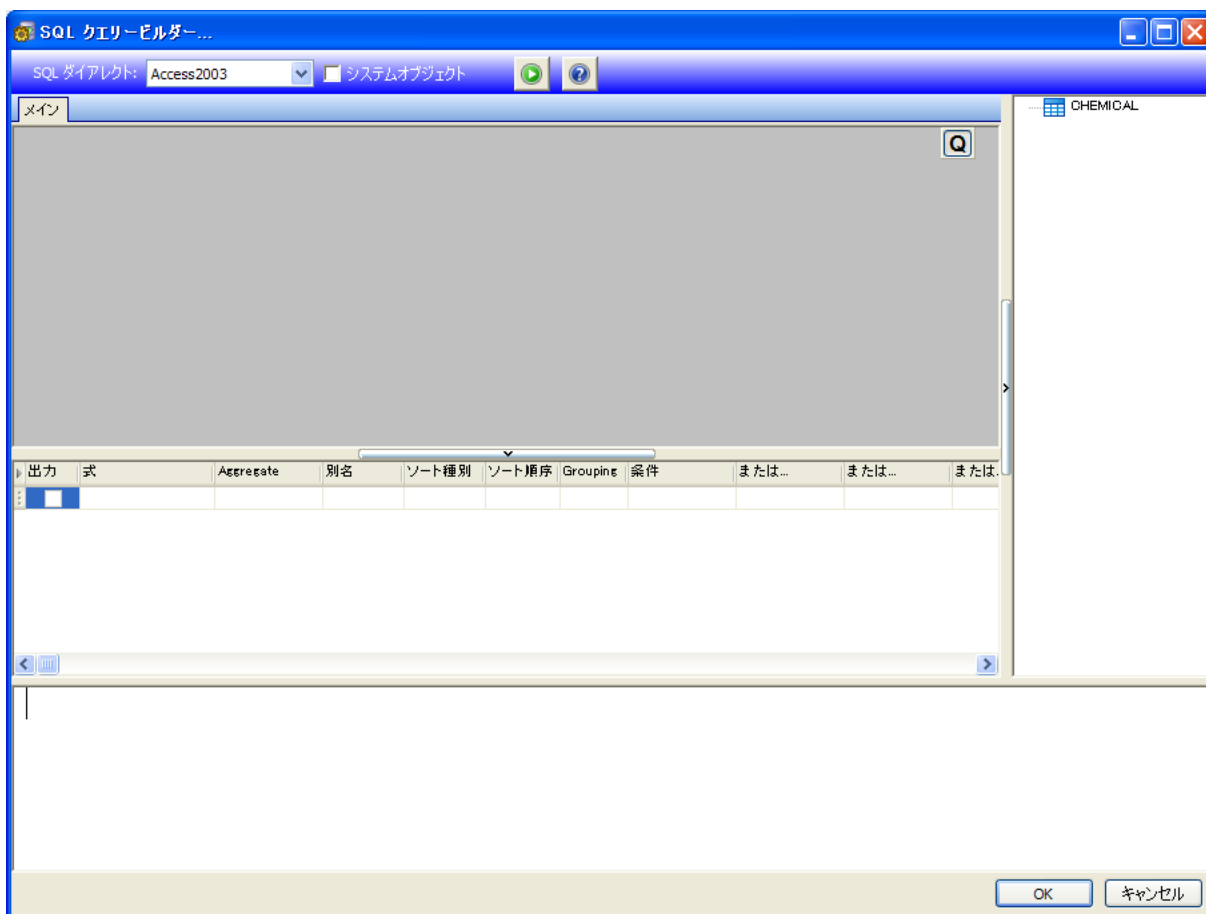
クエリービルダーを取り扱うには、SQL の概念に関する基本知識が必要です。クエリービルダーを使用すると、専門的で詳細な知識がなくても、正しい SQL コードを記述することができます。SQL の原則を理解していれば、望ましい結果を得ることができるでしょう。

起動

クエリービルダーを起動すると、以下の要素が表示されます。

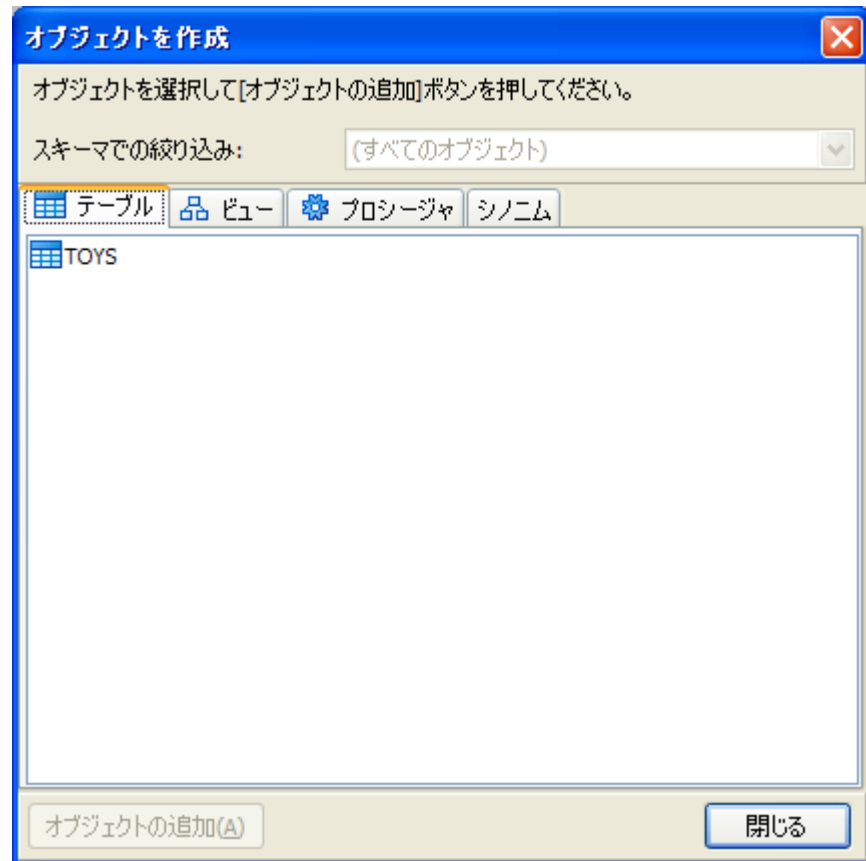
メインウィンドウは、以下の部分に分類されます。

- **クエリー構築エリア** は、クエリーがビジュアルに表示されるメインエリアです。このエリアを使用すると、ソースデータベース オブジェクトと派生するテーブルの定義やそれらのリンクの定義、テーブルやリンクのプロパティを設定することができます。
- **カラムペイン**は、クエリー構築エリアの下にあります。これを使用すると、クエリー出力カラムや式に関する、必要なすべてのオペレーションを行うことができます。こちらで、フィールド別名の定義、ソートやグループ化、条件の定義を行うことができます。
- **クエリーツリーペイン** は右側にあります。こちらで、クエリーをブラウズし、必要な部分をすばやく見つけることができます。
- クエリー構築エリアの上にあるページコントロールを使用すると、メインクエリーとサブクエリーを切り替えることができます。
- クエリー構築エリアの端にある、Q の文字がついた小さなエリアは、結合サブクエリーの取り扱いをコントロールします。こちらで、新しい結合サブクエリーを追加したり、それらに関する必要なオペレーションをすべて行うことができます。



オブジェクトをクエリーに追加

オブジェクトをクエリーに追加するには、クエリー構築エリアを右クリックし、ドロップダウンメニューで**オブジェクトの追加**を選択します。



オブジェクトを作成 ウィンドウを使用すると、複数のオブジェクトを一度に追加することができます。オブジェクトは、タイプによって、**テーブル**、**表示**、**プロシージャ**(ファンクション)および**シノニム** の4つのタブにグループ化されます。**Ctrl** キーを押しながら、1つまたは複数のオブジェクトを選択し、**オブジェクトの追加** ボタンを押すと、これらのオブジェクトをクエリーに追加することができます。この操作を数回繰り返すこともできます。オブジェクトの追加を完了したら、**閉じる** ボタンを押し、このウィンドウを隠します。

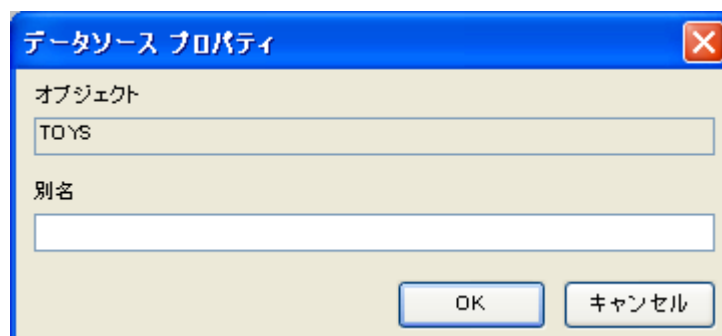
クエリーからオブジェクトを削除するには、オブジェクトを選択し、**Del** キーを押すか、またはオブジェクトのヘッダーにある**閉じる** ボタンをクリックするだけです。

スキーマを持つ、あるいはさまざまなデータベースからオブジェクトを選択できるサーバーについては、ウィンドウ上部にあるコンボボックスから、必要なスキーマまたはデータベースを選び、データベースまたはスキーマの名前別にオブジェクトにフィルターをかけることができます。

クエリービルダーは、データベースで外部キーに関する情報に基づき、テーブル間のリンクを確立することができます。この機能はデフォルトで有効です。この機能をオフにするには、外部キーからのリンクを作成するチェックボックスのチェックを外します。

オブジェクト・プロパティの編集

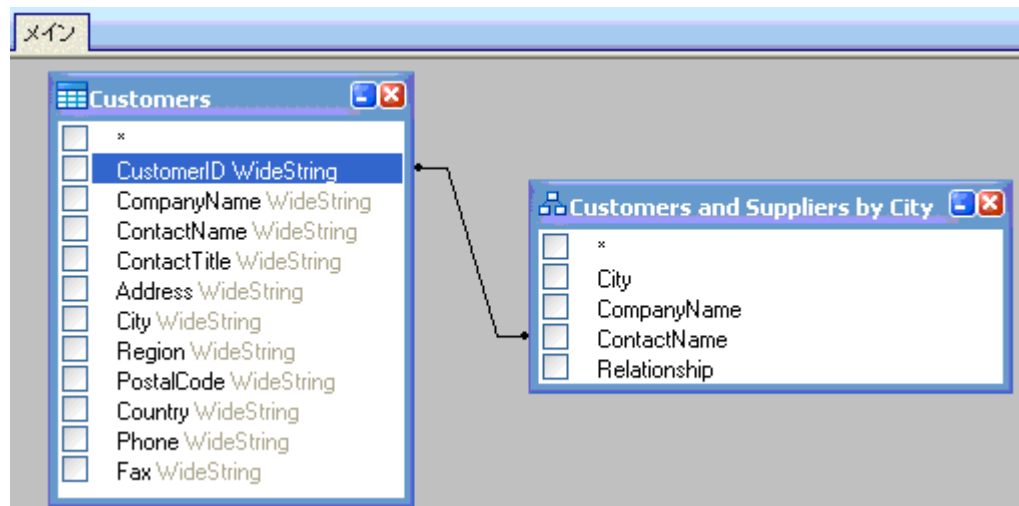
オブジェクトを右クリックし、ドロップダウンメニューから編集を選択するか、オブジェクトのヘッダーをダブルクリックすると、クエリーに追加された各オブジェクトのプロパティを変更することができます。



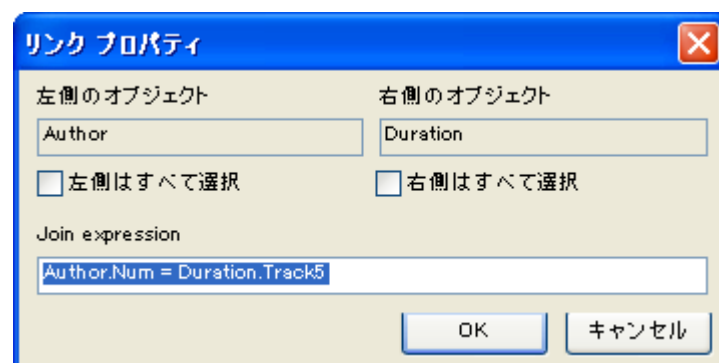
データソース・プロパティダイアログは、サーバーによって異なる場合がありますが、少なくとも別名プロパティはすべてのデータベース・サーバーで同じです。

テーブルの結合

2つのオブジェクト間でリンクを作成(すなわち、結合)するには、オブジェクトのリンクしたいフィールドを選択し、それを、もうひとつのオブジェクトの対応するフィールドにドラッグします。ドラッグを終えると、リンクされたフィールド間を結ぶラインが表示されます。



デフォルトで作成される結合タイプは、INNER JOIN です。つまり、両方のテーブルで一致するレコードのみが、結果のデータセットに含まれます。他の結合タイプを定義するには、リンクを右クリックし、ドロップダウンメニューから編集を選択するか、またはそれをダブルクリックして、リンクプロパティダイアログを開きます。このダイアログを使用すると、結合タイプや他のリンクプロパティを定義することができます。



オブジェクト間のリンクを削除するには、リンクのラインを右クリックし、ドロップダウンメニューで削除を選択してください。

出力フィールドのソート

出カクエリーフィールドのソートを有効にするには、**カラムペインのソートタイプ** や **ソート順**を使用します。

ソートタイプのカラムを使用すると、フィールドのソート方法を、**昇順**または**降順**に指定することができます。

ソート順のカラムを使用すると、1つ以上のフィールドをソートする場合、フィールドのソート順を設定することができます。

特定のフィールドのソートを無効にするには、このフィールドで**ソートタイプ**をクリア(空)にします。

出力	式	Aggregate	別名	ソート種別	ソート順序	Grouping	条件
<input checked="" type="checkbox"/>	Author.Num			昇順	1	<input type="checkbox"/>	
<input checked="" type="checkbox"/>	Author.Author			昇順		<input type="checkbox"/>	
<input type="checkbox"/>				降順			

条件の定義

カラムペイン にリストされる式に関する条件を定義するには、**条件**の列を使用します。

ここで、式自体を省略する条件を記述します。クエリーで下記の条件を取得する場合

WHERE (field >= 10) AND (field <= 20)

条件 に次のように記述します。

>= 10 AND <= 20

または...カラムを使用して、1つの式について複数の条件を指定することができます。これらの条件は、**OR** 演算子を用いて、クエリーに関連付けられます。

パラメータ化したクエリーの定義

クエリービルダーでは、パラメーターの値を変数で与えるパラメータ化したクエリーを作成することができます。

注意: パラメーターを定義する前に変数を作成しておく必要があります。

1. クエリーで使用するテーブルをドラッグ & ドロップします。
2. 条件が適用されるフィールドを選択します。
3. 条件の列、または SQL エディターで、検索条件のオブジェクトとして使用される変数を指定します。


例: 変数 Var0 に格納されている値で検索する場合。

- **SQL エディター:**


```
SELECT [Table].*  
FROM [Table]  
WHERE [Table].Field=APPLICATION.DOCUMENT.Var0
```

- **条件の列:**

```
=APPLICATION.DOCUMENT.Var0
```

4.  ボタンをクリックしてクエリーの結果を表示します。

クエリー結果グリッド

クエリー結果グリッドにアクセスするには、クエリーの定義ダイアログボックスまたはデータベースブラウザへマージツールバー、あるいはデータソース > データベースメニューにより  ボタンをクリックします。


このグリッドを使用すると、クエリーの結果を表示したり、特定の条件とすべての結果を検索し、対応するラベルを印刷したりすることができます。

クエリーの結果グリッドには、以下の機能が含まれます。


- 検索機能


検索を実行するフィールドを選択できる、検索フィールド


検索する値を入力できる、検索データ

フィールドの任意の場所または最初にある値を検索 

- クエリー結果レコードをブラウズする、ナビゲーション機能

最初のレコード 

前のレコード 

次のレコード 

最後のレコード 

- 結果グリッド

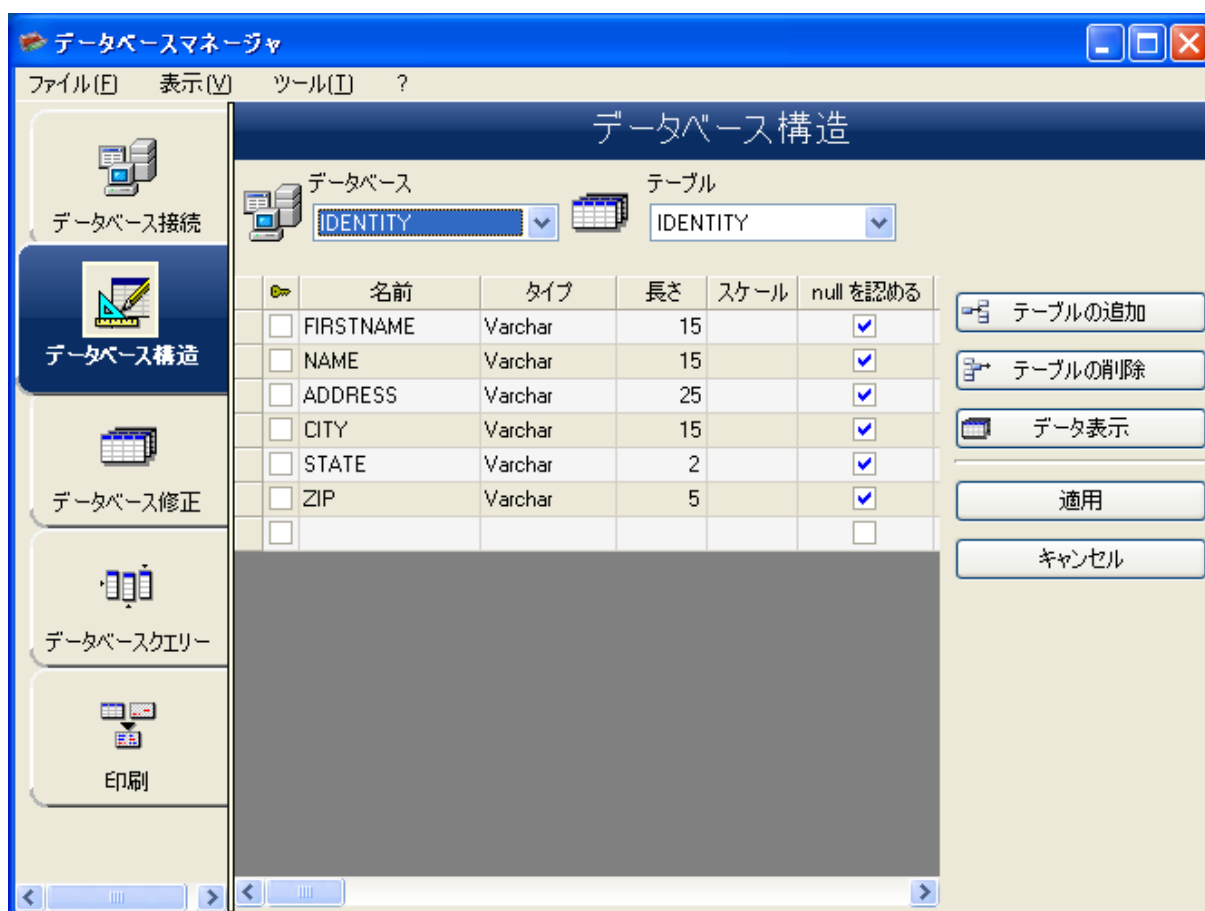
クエリーの実行結果を表示します。

- 再クエリー

リクエストを再度照会し、グリッドを更新します。

データベースマネージャ

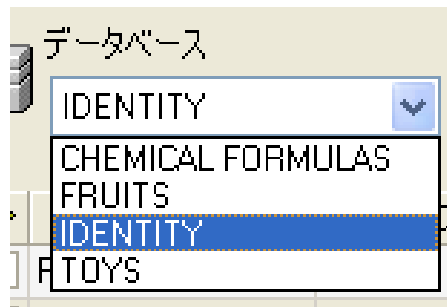
データベース構造ウィンドウ



データベース構造ウィンドウを使用すると、テーブル/フィールドの追加、変更または削除など、データベースのファイル構造を管理することができます。

接続のリストからデータベースを選択

1. データベースのドロップダウンリストをクリックします。
2. 目的のデータベースをクリックします。

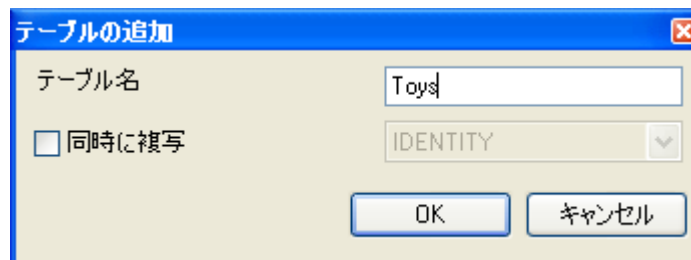


データベースのテーブルを選択

1. テーブルのドロップダウンリストをクリックします。
2. 目的のテーブルをクリックします。

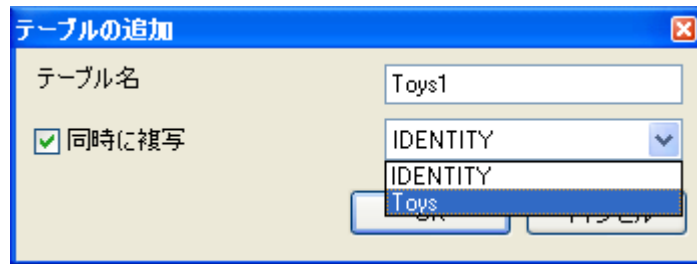
現在のデータベースにテーブルを追加

1. テーブルの追加ボタンをクリックします。
2. 新しいテーブルの名前を入力します。
3. **OK** ボタンをクリックします。



選択されたデータベースにすでに存在するテーブルから、テーブルの構造をコピーすることもできます。この操作方法は以下のとおりです。

1. 「同時に複写」の隣にあるボックスにチェックをつけます。
2. ドロップダウンリストをクリックします
3. 必要なデータをクリックします。
4. **OK** ボタンをクリックします。



現在のデータベースからテーブルを削除

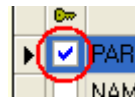
1. テーブルのドロップダウンリストをクリックします。
2. 目的のテーブルをクリックします。
3. テーブルの削除ボタンをクリックします。

現在のテーブルのデータを表示/非表示

1. データ表示ボタンをクリックします。

キーフィールドの定義

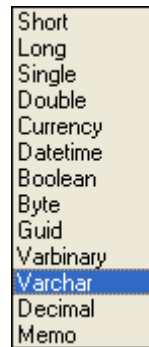
1. 必要なフィールドの隣にあるボックスにチェックをつけます。



2. 適用ボタンをクリックします。

フィールドの内容タイプを定義

1. 対象フィールドのタイプカラムをクリックします。
2. ドロップダウンリストのボタンをクリックします。
3. 希望するデータタイプをクリックします。



4. **適用**ボタンをクリックします。

フィールドの最大サイズを定義

1. 対象フィールドの**長さ**カラムをクリックします。
2. 希望する長さを入力します。
3. **適用**ボタンをクリックします。

空フィールドを許可

1. 対象フィールドの「**Null を認める**」カラムにチェックをつけます。
2. **適用**ボタンをクリックします。

データベース修正ウィンドウ



データベース修正ウィンドウを使用すると、データの追加、変更または削除など、データベースファイルの内容を管理することができます。

これらのアクションは、データベースのタイプによって異なります。このため、Excel ファイルのレコードは修正できません。


内容によりレコードを選択

フィールドの内容を使用して、レコードを検索します。

1. 検索フィールドの選択ドロップダウンリストのボタンをクリックします。
2. 目的のフィールドをクリックします。
3. データ入力フィールドをクリックします。
4. データ入力フィールドに検索する値を入力します。

一致するレコードをすべて選択

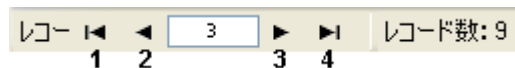
1つ以上のレコードが表示されていなければなりません。

1. **検索フィールドの選択**ドロップダウンリストのボタンをクリックします。
2. 目的のフィールドをクリックします。
3. データ入力フィールドをクリックします。
4. データ入力フィールドに検索する値を入力します。
5. **全てを選択** () ボタンをクリックします。

一致するレコードを選択

1つ以上のレコードが表示されていなければなりません。また、検索フィールドにいくつかの一致する内容があります。

レコードを選択するには、**検索ツール**を使用します。1(先頭レコード)、2(前レコード)、3(次レコード)または4(最終レコード)をクリックします。



新しいレコードの作成

1. アスタリスク(*)が付いた行のフィールドをクリックします。
2. 対応するフィールドに必要な値を入力します。
3. **適用**ボタンをクリックします。

レコードの変更

1. 変更したいデータをクリックします。
2. 必要なデータを入力します。

3. **適用**ボタンをクリックします

レコードの削除

1. 目的のレコードのデータベースのカーソルをクリックします。
2. 右クリックします。
3. コンテキストメニューで、**選択レコードの削除**をクリックします。

データベースクエリーウィンドウ



データベースクエリーウィンドウを使用すると、さまざまなフィルターを作成し、適用することができます。

クエリーの追加

1. クエリーの追加ボタンをクリックします。
2. クエリーの名前を入力します。
3. **OK** ボタンをクリックします。

1つまたは複数のフィールドを選択/選択解除


1. ナビゲーションツール  を使用します。

2. クエリーボタンをクリックし、データベースプレビューを更新します。


選択したフィールドの順序を変更

1. 並べ替えフィールドリストで目的のフィールドをクリックします。
2. 上下の矢印ボタンをクリックして、フィールドの順序を調整します。
3. クエリーボタンをクリックし、データベースプレビューを更新します。

事前に定義されたデータを使用し、フィルターを作成


1. フィルター タブを選択します。
2. レコードの追加ボタン()をクリックします。
3. フィールドにおいて、ドロップダウンリストから希望するフィールドを選択します。
4. 演算子 フィールドにおいて、ドロップダウンリストから希望する演算子を選択します。
5. 値 フィールドにおいて、条件となる値を入力します。
6. クエリー ボタンをクリックし、結果を表示します。

論理演算子を複数のフィルターに適用

1. レコードの追加ボタン ()をクリックします。
2. 論理式 フィールドにおいて、ドロップダウンリストから AND または OR を選択します。
3. フィルターを作成します (上記に記載)。
4. クエリー ボタンをクリックし、結果を表示します。

フィルターの削除

注意:1 つ以上のフィルターが必要です。

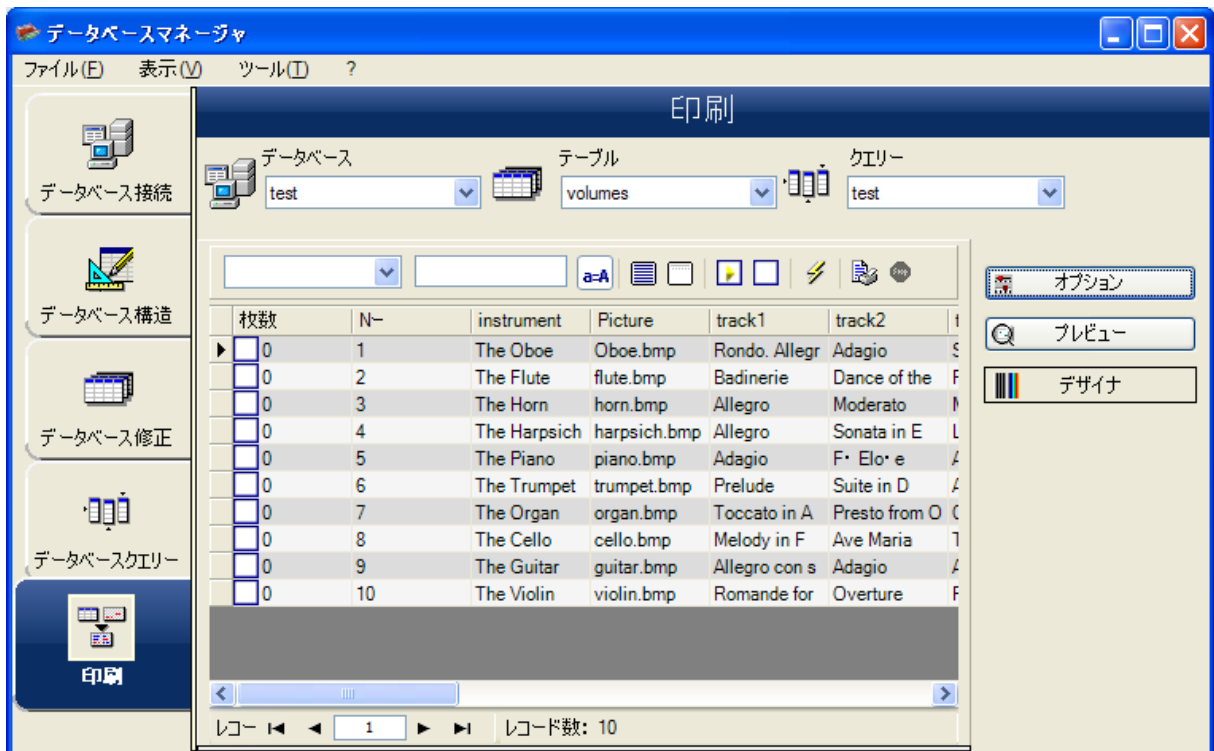
1. 目的のフィルターのデータベースのカーソルをクリックします。
2. レコードの削除ボタン()をクリックします。

SQL でフィルターを変更

注意:1 つ以上のフィルターが必要です。

1. **SQL クエリー**タブを選択します。
2. 「**SQL 文で編集**」にチェックをつけ、SQL クエリーを有効にして、手動で変更を行います。
3. **クエリー**ボタンをクリックし、結果を表示します。

印刷ウィンドウ




印刷ウィンドウを使用すると、印刷を開始する前に、印刷するファイルの選択、プリンターの指定、さまざまなパラメーターの定義を行うことができます。

印刷するドキュメントを選択

1. オプションボタンをクリックします。
2. ラベル名グループのファイルを選択します。
3. ファイルからドキュメントを選択します。

または

1. ラベル作成アシスタントボタン () をクリックします。
2. ウィザードの説明にしたがって、進めます。

注意: データベースに関連するラベルを作成すると、データベースの各フィールドの位置づけに必要な要素を、正確に定義することができます。

既存のラベルを選択

1. 既存のラベルドキュメントを開くボタン (📄) をクリックします。
2. .lab ファイルを選択します。
3. 開くボタンをクリックします。

注意: オプションの「ラベル名」や「プリンター名」グループにある「フィールド」ラジオボタンを使用すると、現在のデータベースのフィールドのひとつにそれらの文字が定義されている場合に、必要なラベル、プリンターが選択されます。

フィールドからドキュメントを選択

データベースにおいて、フィールドのひとつに印刷ジョブに使用されるラベル名が含まれる場合、このフィールドを、データベースマネージャーが.lab ファイルを取得する場所として定義することができます。

Ref	Designation	Qt	Code	Labname
6574	Ref1	1	9876546321	Label1.lab
6354	Ref2	2	1236478855	Label2.lab
6987	Ref3	3	6987456321	Label1.lab
3684	Ref4	4	3698745632	Label3.lab

この例では、フィールド「Labname」は、Labname フィールドとして使用できません。

1. ラベル名グループのフィールドを選択します。
2. 必要なフィールドを選択します。

プリンターの選択

プリンターの追加と削除ボタンをクリックします。

数式データソース

コマンド: データソース > 数式 > 追加

数式データソースには、作成したデータソースのリストが含まれます。これらのデータソースには、演算子、定数、データソース、予約変数、式および関数の組み合わせが存在しています。データは数字または英数字となります。

ドキュメント内で演算を行う場合、まず、数式データソースを作成しなければなりません。

このデータソースには、特定の式に関して必要な関数を定義できるダイアログボックスがあります。

関数について

関数とは、事前に定義された数式のことです。引数と呼ばれる特定の値を、シンタックスと呼ばれる特定の順番で使用して、演算を実行します。

関数は演算またはオペレーションの結果として、数字、文字列または論理値を返します。

数式の定義では、以下の6つの関数グループがあります。

- チェックデジット関数
- 型変換関数
- 日時と時刻
- 論理関数
- 数学関数

- 文字関数

演算子

演算子は、実行するオペレーションを示す数学記号です。算術、比較、連結、論理など、さまざまなタイプの演算子があります。

算術演算子

演算子	用途
*	乗算: 2つの数を掛け合わせます。
+	加算: 2つの数を加算します。
-	減算: ある数から別の数を引くか、あるいはオペランドにマイナス値を指定します。
/	除算: ある数を別の数で割ります。
^	べき乗: ある数を指数で累乗します。
%	剰余: ある数を別の数で割った余りです。

比較演算子

演算子	意味
<	~より小さい
<=	~より小さいまたは等しい
>	~より大きい
>=	~より大きいまたは等しい
=	~と等しい
<>	~とは異なる

連結演算子

2つの文字列を組み合わせる場合に使用します。

演算子	意味
&	2つの文字列の連結

論理演算子

(論理関数も参照してください)

演算子	意味
!	論理否定

数学関数

base10tobaseX(*string_1*, *string_2*) : *string_2* を 10 進数 から *string_1* 進数へ変換します。

例:

Base 16 という名前のフィールドに文字列「0123456789ABCDEF」が含まれる場合、

base10tobaseX(Base16, 12) は C を生成します。

base10tobaseX(Base16,10) は A を生成します。

base10tobaseX("012345","9")は 13 を生成します。

注意:この式は、「string_2」引数に負の値をとることができません。

baseXtobase10(*string_1*, *string_2*) : *string_2* を *string_1* 進数から 10 進数へ変換します。

例:

Base 16 という名前のフィールドに文字列「0123456789ABCDEF」が含まれる場合

baseXtobase10(Base16, "E") は 14 を生成します。

baseXtobase10(Base16,10) は 16 を生成します。

baseXtobase10("012345",10)は 6 を生成します。

eval_add(*string_1*, *string_2*) : パラメーターの加算を返します。

例:

eval_add(5, 5) = 10

eval_div(string_1, string_2) : パラメーターの除算を返します。

例:

eval_div(20, 2) = 10

eval_mult(string_1, string_2) : パラメーターの乗算を返します。

例:

eval_mult(5, 2) = 10

eval_sub(string_1, string_2) : パラメーターの減算を返します。

例:

eval_sub(20, 10) = 10

hex(val_1, val_2) : 10 進数の val_1 を val_2 桁の 16 進数に変換します。

注意:: この式は、「val_1」、「val_2」引数に負の値をとることができません。

int(値) : 値 引数より小さいまたは等しい、最大整数を返します。

例:

int(-5.863) = -6

int(5.863) = 5

mod(val_1, val_2) : val_1 引数を val_2 引数で割った余りを返します。符号は被除数と同じになります。

例:

mod(7,2) = 1

mod(-7,2) = -1

mod(7,-2) = 1

mod(-7,-2) = -1

quotient(val_1, val_2) : val_1 引数を val_2 引数で割った商を整数で返します。

round(val_1, val_2) : val_1 引数 を、val_2 の桁数で四捨五入した値を返します。

- val_2 が 0 より大きい場合、val_1 は、小数点以下の桁数で丸めます。
- val_2 が 0 に等しい場合、val_1 は最も近い整数で丸めます。
- val_2 が 0 より小さい場合、val_1 は、小数点の左側の桁数で丸めます。

例:

round(4.25,1) = 4.3

round(1.449, 1) = 1.4

round (42.6,-1) = 40

trunc(value) : value 引数の整数部分を返します。

論理関数

論理関数を使用すると、1 つまたは複数の条件を満たしたかどうかをチェックすることができます。

注意: TRUE は 1、FALSE は 0 となります。

and(expr_1, expr_2) : 両方の引数が正しい場合、TRUE を返します。少なくとも 1 つが正しくない場合、FALSE となります。引数は、論理値から計算しなければなりません。

exact(string_1, string_2) : 2 つの文字列が同じ場合 TRUE、異なる場合は FALSE を返します。この関数は大文字・小文字を区別します。

例:

exact("String","String") = 1

exact("String","string") = 0

if(expr, Val_if_true, Val_if_false) : expr が正しい場合は Val_if_true 値を返し、expr が正しくない場合は Val_if_false 値を返します。

例:

```
if(exact("String", "string"), "true", "false") = false
```

```
if(exact("String", "String"), "true", "false") = true
```

not(*logical*) :: 論理式 *logical* の否定を返します。

例:

```
not(exact("String", "string")) = 1
```

```
not(exact("String", "String")) = 0
```

```
not(False) = 1 または not(0) = 1
```

```
not(True) = 0 または not(1) = 0
```

```
not(1+1=2) = 0
```

or(*expr_1*, *expr_2*) : 2つの引数の内1つが正しい場合 TRUE を返し、引数が両方とも正しくない場合 FALSE を返します。この引数は論理値から計算しなければなりません。

例:

```
or(exact("String", "string"), exact("string", "String")) = 0
```

```
or(exact("String", "String"), exact("string", "String")) = 1
```

```
or(true, true) = 1 または or(1, 1) = 1
```

```
or(true, false) = 1 または or(1, 0) = 1
```

```
or(false, false) = 0 または or(0, 0) = 0
```

文字関数

各ボックスに文字が含まれている場合、文字列をテーブルに含めることができます。それは長さで定義します(スペースを含む、文字列の総文字数)。文字列の文字の位置は、テーブルの場所に対応します。例えば、1文字目は位置1になります。

例: 位置 3 は文字列の 3 番目の文字に対応します。

cyclebase(*baseX_string*, *start_value*, *increment*, *copies*) : あらゆるデータベース・カウントシステムにおいて、カウントを利用可能とします。ナン

バリングシステムは、リンクされた式内で定義しなければいけません。開始値、それぞれの増分の値、コピー数もそれぞれの数について指定します。これらの値はすべて、ラベルの他のフィールドにリンクさせることができますが、フィールド名を引用符に含めてはいけません。

例:

Base 16 という名前のフィールドに文字列 0123456789ABCDEF が含まれる場合、以下のとおりです。

`cyclebasex(base16, "8", 1, 1) = 8,9,A,B,C...`

`cyclebasex(base16, "F", -1, 1) = F,E,D,C,B,A 9,8,7...`

`cyclebasex(base16, "B0 ", 1, 1) = B0, B1, B2...`

`cyclebasex("012345", "4", 1, 2) = 4,4,5,5,10,10,11,11...`

`cyclechar(first_char, last_char, increment, copies)` : 完全なサイクルについて、ユーザー定義の文字セットを作成します。

例:

`cyclechar("A", "C") = A B C A B C A B C ...`

`cyclechar("A", "C", 1, 2) = A A B B C C A A B B ...`

`cyclenumber(first_char, last_char, increment, copies)` : 通常の数や文字のシーケンスを使用するのではなく、独自の数字のシーケンスを設定することができます(0,1,2... or A,B,C...).

例:

`cyclenumber(1,3)` は次のシーケンスのラベルを作成します: 1 2 3 1 2 3
1 2 3...

`cyclenumber(1,3,1,2)` は次のシーケンスのラベルを作成します: 1 1 2 2
3 3 1 1 2 2 3 3 ...

`cyclestring(string)` : 増分フィールドとして、完全なサイクルを使用し、単語または文字のグループを作成することができます。この完全な文字列は引用符(" ")に含めなければならず、各単語または文字グループはセミコロン(;)で区別しなければなりません。

例:

```
cyclestring("Mon ; Tue ; Wed ; Thu ; Fri ; Sat ; Sun") = Mon Tue  
Wed Thu Fri Sat Sun
```

以下の例は、O と I を除く、すべてのアルファベット文字を使用するラベルを示します。

```
cyclestring("A;B;C;D;E;F;G;H;J;K;L;M;N;P;Q;R;S;T;U;V;W;X;Y;Z")
```

exact(*string_1*, *string_2*) : 2 つの文字列が同じ場合 TRUE、異なる場合 FALSE を返します。

例:

```
exact("software","software") = 1  
exact("sftware","software") = 0
```

extract(*string*, *sep*, *pos*) : 文字列 *sep* で区別されるデータを含む文字列 *string* から位置 *pos* の文字列を返します。

例:

```
Extract("AB;CD;EFG;HIJ", ";", 3) = "EFG"
```

find(*string*, *key*, *start*) : *string* 引数で最初に現れる *key* 引数の位置を返します。*string* 引数の検索は、*start* 引数 (*start* >= 1) によって返される位置から始まります。*key* 引数が見つからない場合は、この関数はゼロを返します。この関数は、大文字・小文字を区別します。

例:

```
find("Peter McPeepert","P",1) = 1  
find("Peter McPeepert","p",1) = 12
```

left(*string*, *num_char*) : 文字列 *string* の先頭から *num_char* 引数で与えられた長さの文字列を返します。

例:

```
left("Peter McPeepert",1) = P  
left("Peter McPeepert ",5) = Peter
```

len(*string*) :: *string* 引数の長さを返します。スペースも文字としてカウントします。

例:

```
len("Paris, New York") = 15
```

```
len("") = 0
```

```
len(" ") = 1
```

lower(*string*) :: 文字列 *string* をすべて小文字に変換します。

例:

```
lower("Paris, New York") = paris, new york
```

mid(*string*, *start*, *num_char*) : 文字列 *string* の *start* 引数 (*start* >=1) で与えられた位置から *num_char* 引数で与えられた長さの文字列を返します。

例:

```
mid("Paris, New York",8,8) = New York
```

pad(*string*, *length*, *char*) : 指定された長さになるように、フィールドの左側に指定した文字を追加します。すべての文字をパディング文字として選択できます。

例:

REETING という名前のフィールドに HELLO の値を表示する場合、

```
pad(GREETING,8,0) = 000HELLO
```

```
pad(5,3,0) = 005
```

```
pad("Nine",6,"a") = aaNine
```

replace(*string*, *start*, *num_char*, *new_string*) : 変換された *string* 引数を返します。*start* 引数で与えられた位置から *num_char* 引数で与えられた文字を *new_string* 引数で与えられた文字列に置換します。

例:

```
replace("Paris, New York",8,8,"Singapore") = Paris, Singapore
```

replaceString(string, old_string, new_string) : 文字列 *string* 内のすべての *old_string* を *new_string* に置換します。

例:

```
ReplaceString( "abc12def12", "12", "" ) = abcdef
```

rept (string, num_char) : *string* 引数を *num_char* 回繰り返した文字列を返します。

例:

```
rept("Ah Paris! ",2) = Ah Paris! Ah Paris!
```

right (string, num_char) : 文字列 *string* の最後の文字から *num_char* で与えられた長さの文字列を返します。

例:

```
right("Purchase order",5) = order
```

search (string, key, start) : 文字列 *string* 内で *start* (*start* >= 1)位置以降に現れる *key* の位置を返します。検索は、*start* 引数 (*start* >= 1)により定義される位置から始まります。この関数は、*key* 引数が見つからない場合、ゼロを返します。

例:

```
search("Purchase order","order",1) = 10  
search("Purchase order","c",1) = 4
```

trim (string) : 文字列 *string* の始めと終わりのすべてのスペースを削除します。また単語間に複数のスペースが存在する場合は、1つに削減します。

例:

```
trim(" Purchase order") = Purchase order
```

trimall (string) : 文字列 *string* からすべてのスペースを削除します。

例:

```
trimall("Paris / New York / Rome") = Paris/NewYork/Rome
```

upper (string) : 文字列 *string* をすべて大文字に変換します。

例:

```
upper("Purchase order") = PURCHASE ORDER
```

ztrim () : 完全な数字フィールドのすべてのゼロを左から削除します。

例:

WEIGHT という名前のフィールドが 000200 の値を表す場合

```
ztrim(weight) = 200
```

数式データソースのプロパティを定義

コマンド: データソース > 数式 > 追加

1. **編集ボックス**に直接、数式を入力します

または

必要な要素を選択し、**挿入ボタン**をクリックします。

2. **テストボタン**をクリックすると、シンタックスが正しいかどうかを確認できます。エラーが発生した場合、画面の指示に従って、必要な変更をすべて行ってください。

3. **OK ボタン**をクリックします。

ヒント: ダブルクリックにより要素を挿入することができます。

注意: 数式に使用される変数に、以下の文字の1つを含む名前がある場合、それを括弧 {} に含めなければいけません。

```
&+ - * / < > = ^ % , ! \"
```

注意: テストボタンをクリックすると、数式をチェックすることができます。メッセージボックスに結果の値が表示された場合は式が正しいことを表します。式が正しくない場合は、画面の指示に従って必要な変更を行ってください。結果の値が切れている場合は出力タブで指定している最大長を変更してください。

実習: 簡単な式の作成

製品の価格を表示

生産ラベルには、重量と1キロあたりの価格を関数処理した製品の価格を表示します。

1. ラベルを作成します。「WEIGHT」と「PRICEPERKG」の2つの変数を作成しておきます。
2. WEIGHT 変数: 変数のローカル値として788 (製品の重量788g)を入力し、出カタブの接頭文字入力欄に「重量をグラムで入力してください:」を入力し、OK ボタンをクリックします。
3. PRICEPERKG 変数: 変数のローカル値として15.70 (価格/kg FF15.70)を入力し、出カタブの接頭文字入力欄に「1キロあたりの価格を入力してください:」を入力して、OK ボタンをクリックします。
4. 数式変数を作成し、「Price」と名付けます。
5. 数式: 「WEIGHT*PRICEPERKG/1000」を入力し、OK ボタンをクリックします。(注: 前後の「」は数式には含めません。)
6. ラベルファイルを保存します。

実習: 警告メッセージを表示するための「Warning」数式変数を追加

下記のシーケンスでは、ユーザーに警告メッセージを表示し、Total_Weight 共有変数の値が1,000kgを超えていることを伝える場合の式を作成します。

重量の値が1,000kgを超える場合、「ご注意ください！エラー！総重量が最大値を超えています！」というメッセージを表示します。

1. ラベルを作成します。「Total_Weight」という変数を作成しておきます。

2. 数式変数を作成し、「Warning」と名付けます。
3. **数式** のダイアログボックスにおいて、以下の式を入力します。
「if(Total_Weight>1000, "ご注意ください！\n エラー！\n 総重量が最大値を超えています！", "")」(注: 前後の「」は数式には含めません。)
4. **出力タブ**では、**最大の長さ**に 50 を入力し、**OK** ボタンをクリックします。
5. 変数をラベル内のテキストとして設定します。
6. **パラグラフ** タブでは、**自動行送り** オプションにチェックをつけ、位置合せで**中央揃え**を選択します。

IF 関数に関する情報

指定された条件が TRUE の場合は、ある値を返し、FALSE の場合には、別の値を返します。

IF 関数を使用して、値や式に関する条件テストを行います。

シンタックス

if(*expr*, *val_if_true*, *val_if_false*) : *expr* が値または式を表す場合、その結果は TRUE または FALSE

val_if_true は、*expr* が TRUE の場合に返される値です。*val_if_true* 引数は別の式とすることもできます。

val_if_false は、*expr* が FALSE の場合に返される値です。*val_if_false* 引数は別の式とすることもできます。

実習: 特別なチェックデジットの作成

この例では 2/5 interleaved バーコード用のチェックデジットを計算します。

ウエイトの計算

チェックデジットはデータの最初の文字に 1 を掛け、2 番目の文字は 2、3 番目の文字は 1、4 番目の文字は 2...を掛けて算出します。

チェックデジット文字は Customer_Code 変数の値から計算されます。予め Customer_Code 変数を作成し、値として 26053 を入力しておきます。

Formula_1_Weighted 式を作成し次の式を入力します。

(注:ここでは見やすいように改行していますが、実際の入力では改行は不要です。)

Formula_1_Weighted:

```
mid(Customer_Code, 1, 1) * 1 &  
mid(Customer_Code, 2, 1) * 2 &  
mid(Customer_Code, 3, 1) * 1 &  
mid(Customer_Code, 4, 1) * 2 &  
mid(Customer_Code, 5, 1) * 1
```

Customer_Code の値が **26053** ですので

$$2 * 1 = 2$$

$$6 * 2 = 12$$

$$0 * 1 = 0$$

$$5 * 2 = 10$$

$$3 * 1 = 3$$

となり、結果を連結して「**2120103**」が得られます。

ウエイトの計算結果を加算

次のステップでは、前の式から導き出された数値を加算します。この文字列の最大許容長さは 2 であることにご留意ください。Formula_2_Sum 式を作成し次の式を入力します。(注:ここでは見やすいように改行していますが、実際の入力では改行は不要です。)

Formula_2_Sum:

```
mid(Formula_1_Weighted, 1, 1) +
mid(Formula_1_Weighted, 2, 1) +
mid(Formula_1_Weighted, 3, 1) +
mid(Formula_1_Weighted, 4, 1) +
mid(Formula_1_Weighted, 5, 1) +
mid(Formula_1_Weighted, 6, 1) +
mid(Formula_1_Weighted, 7, 1)
```

結果として「 $2 + 1 + 2 + 0 + 1 + 0 + 3 = 9$ 」が得られます。

チェックデジットの計算

前の結果を用いて、チェックデジットの値を計算します。

Formula_3_CheckDigit 式を作成し次の式を入力します。

Formula_3_CheckDigit:

```
if ((Formula_2_Sum % 10) > 0, 10 - Formula_2_Sum % 10, 0)
```

この計算は、先に計算したウエイト加算値 9 を 10 で割り、その余りが 1 以上なら 10 から余りを引きます。この例では余りが 9 なので、

$$10 - 9 = 1$$

この操作はチェックデジット文字が 2 桁になった場合に、1 桁に戻すためのトリックです。

エンコードするデータの計算

バーコードを作成する場合、エンコードするデータを含まなければなりません。

例えば、チェックデジットの値(Formula_3_CheckDigit)に連結される Customer_Code 変数の値などです。

4 番目の式を作成し、Formula_4_NewCustCode と名前を付けます。この式は、Customer_Code と Formula_3_CheckDigit を連結した結果となります。式は以下のとおりです。

Formula_4_NewCustCode:
Customer_Code & Formula_3_CheckDigit

結果として

26053 & 1 = 「**260531**」

が得られます。

バーコードの作成

1. Formula_4_NewCustCode 式を選択し、Customer_Code バーコードのラベルにドラッグ & ドロップします。
2. バーコードのプロパティを定義します。



United States
1-414-837-4800

France
33-562-601-080

Germany
49-6103-30026-0

Asia
65-6570-4923

Japan
81-45-461-3603

Copyright 2013 Teklynx Newco SAS. All rights reserved. TEKLYNX, CODESOFT, LABEL ARCHIVE and SENTINEL are trademarks or registered trademarks of Teklynx Newco SAS or its affiliated companies. All other brands and product names are trademarks and/or copyrights of their respective owners.

www.teklynx.com | www.teklynx.eu

