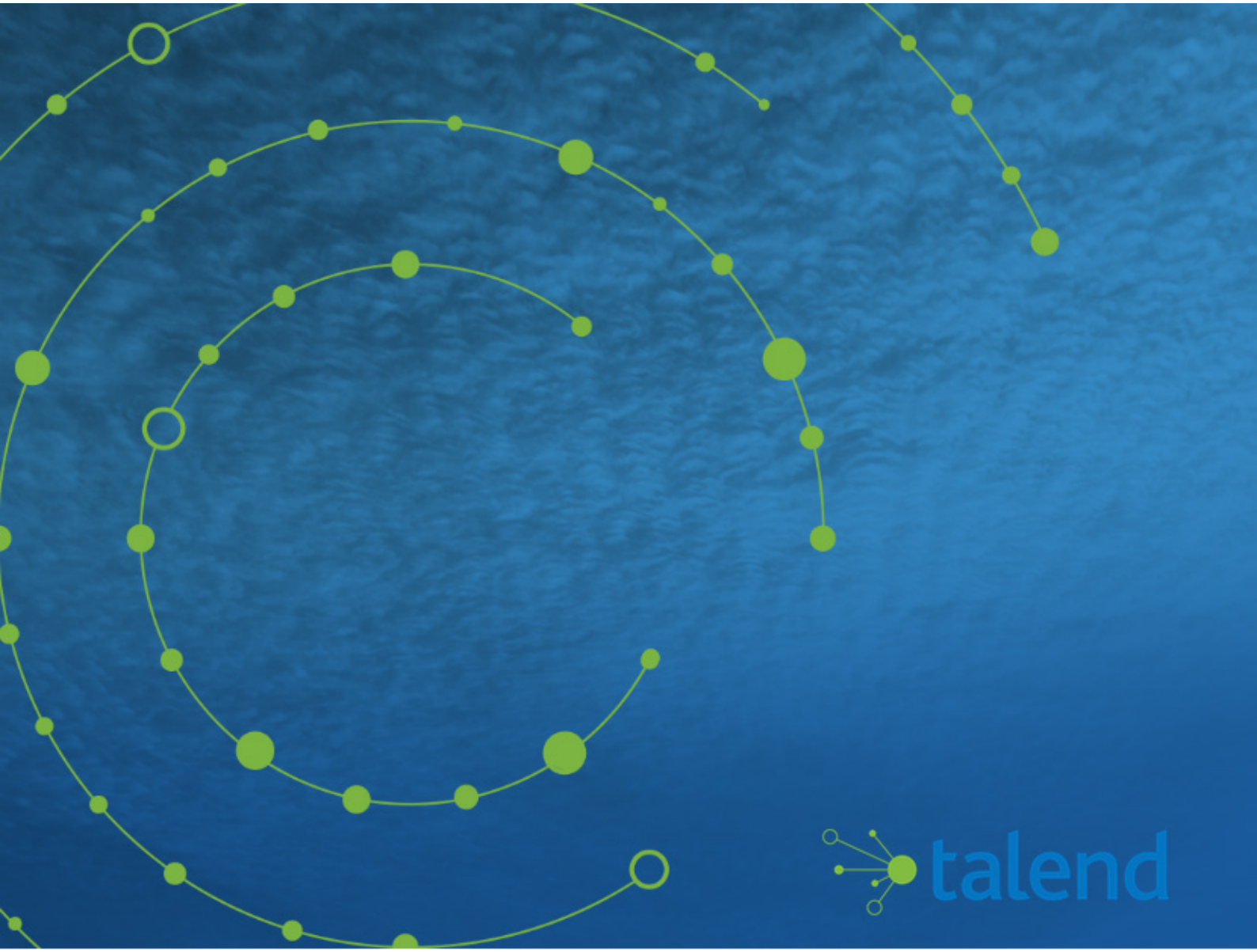


Data Integrationジョブの例

7.0.1

目次

Copyleft

7.0.1に適合。 以前のリリースの更新版となります。

公開日:2018年4月13日

注意

ライセンス契約

tMapジョブの例

以下、実際に起こり得るサンプルシナリオを示すことにより、Talend Studioの操作方法を紹介します。このシナリオでは、ファイルをMySQLテーブルにロードします。ファイルはオンザフライで変換されます。その後、動的フィルターを使用してロードするデータを選択します。

実際にジョブを開始する前に、入力データおよび予想される出力データを確認しましょう。

入力データ

入力ファイル(データベーステーブルにロードされるデータ)には、カリフォルニア州全土の顧客リストが含まれています。

このファイル構造(Talend Studioでは通常**スキーマ**と呼ばれる)には、次のカラムが含まれます。

- First name (名)
- Last name (姓)
- Address (住所)
- City (都市)

出力データ

データベースに、カリフォルニア州の2つの郡(オレンジ郡とロサンゼルス郡)に住んでいる顧客のみをロードします。

テーブル構造が少し異なるため、DBテーブルにロードするデータは、次の構造にする必要があります。

- `Key` (キー、Integer型)
- `Name` (String型、最大長:40)
- `Address` (String型、最大長: 40)
- `County` (String型、最大長:40)

このテーブルをロードするために、次のマッピングプロセスを使用する必要があります。

`Key` (キー)カラムには、自動的にインクリメントされた整数が入力されます。

`Name` (名前)カラムには、姓と名が連結された値が入力されます。

`Address` (住所)カラムには、入力ファイルの対応するAddress (住所)カラムのデータが入力されますが、ロード前に大文字変換が行われます。

`County` (郡)カラムには、オレンジ郡とロサンゼルス郡の都市をフィルタリングするための参照ファイルが使用され、その都市がある郡の名前が入力されます。

参照データ

オレンジ郡とロサンゼルス郡のデータのみをデータベースにロードするため、オレンジ郡とロサンゼルス郡のみがフィルタリングされるように、カリフォルニアの都市をそれぞれの郡にマッピングする必要があります。

そのためには、オレンジ郡とロサンゼルス郡にある都市のリストを含む次のような参照ファイルを使用します。

City (都市)	County (郡)
Agoura Hills (アゴウラヒルズ)	Los Angeles (ロサンゼルス)
Alhambra (アルハンブラ)	Los Angeles (ロサンゼルス)
Aliso Viejo (アリソビエホ)	Orange (オレンジ)
Anaheim (アナハイム)	Orange (オレンジ)
Arcadia (アルカディア)	Los Angeles (ロサンゼルス)

このジョブの参照ファイルの名前は、`LosAngelesandOrangeCounties.txt`です。

シナリオをジョブに変換する

このシナリオを実装するには、ジョブに関する次の4つの手順が必要です。

1. ジョブの作成、入力ファイルパラメーターの設定、入力ファイルの読み取り
2. データのマッピングと変換
3. 参照ファイルパラメーターの定義、**tMap**コンポーネントを使用した関連マッピング、内部結合モードの選択
4. MySQLテーブルへの出力のリダイレクト

手順1: ジョブの作成、入力の定義、ファイルの読み取り

手順

1. Talend Studioを起動してローカルプロジェクトを作成するか、Talend Studioの初回起動時は、デモプロジェクトをインポートします。
2. ジョブを作成するには、**[Repository]** (リポジトリ)ツリービューで**[Job Designs]** (ジョブデザイン)を右クリックし、**[Create Job]** (ジョブの作成)を選択します。
3. 表示されるダイアログボックスでは、最初のフィールド**[Name]** (名前)のみが必須です。**California1**と入力し、**[Finish]** (終了)をクリックします。

   空のジョブがメインウィンドウに表示され、機能別コンポーネントの**[Palette]** (パレット)が(デフォルトではStudioの右側に)現れ、12のコンポーネントファミリー(**Databases** (データベー

ス)、**Files** (ファイル)、**Internet** (インターネット)、**Data Quality** (データクオリティ)など)が表示されます。何百ものコンポーネントが利用可能です。

4. ファイル`California_Clients`を読み取るのに**tFileInputDelimited**コンポーネントを使用します。このコンポーネントは**[Palette]** (パレット)の**[File] (ファイル)** > **[Input] (入力)**グループに含まれています。このコンポーネントをクリックし、さらにデザインワークスペースの左側をクリックして、それをデザインエリアに配置します。
5. このコンポーネントの読み取りプロパティとしてファイルパス、カラム区切り記号、エンコーディングなどを定義します。この操作には**Metadata Manager**を使用します。このツールは、パラメーターを設定するために役立つ多数のウィザードを備えています。設定したプロパティを保存しておくことにより、将来のすべてのジョブで、ワンクリックで再利用できます。
6. 入力ファイルは区切り記号付きフラットファイルであるため、**[Repository]** (リポジトリ)ツリービューの**Metadata**フォルダーを右クリックして、**[File Delimited]** (区切り記号付きファイル)を選択します。次に、**[Create file delimited]** (区切り記号付きファイルの作成)を選択します。

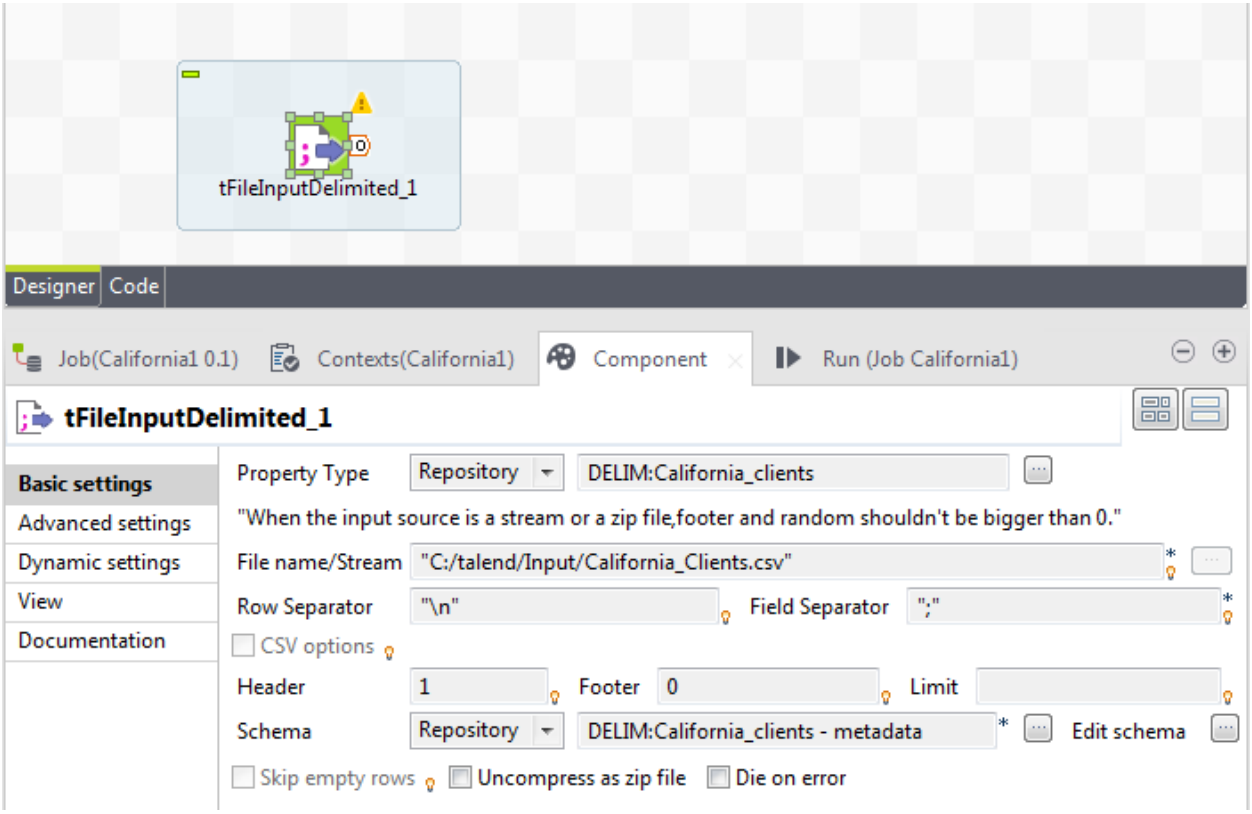
   区切り記号付きファイル専用のウィザードが開き、以下の内容が表示されます。

   - 手順1では**[Name]** (名前)フィールドのみが必須です。**California_clients**と入力して、次の手順に進みます。
   - 手順2で、**[Browse...]** (参照)ボタンを使用して、入力ファイル(`California_Clients.csv`)を選択します。画面下部の[Preview] (プレビュー)にファイルの抜粋が表示されるので、ファイルの内容をすぐに確認できます。**[Next]** (次へ)をクリックします。
   - 手順3で、ファイルパラメーター(ファイルエンコーディング、行とカラムの区切り記号など)を定義します。この入力ファイルは非常に標準的なものなので、大部分がデフォルト値で構いません。ファイルの1行目はカラム名を含むヘッダーです。自動的にこれらの名前を取得するため、**[Set heading row as column names]** (カラム名として先頭行を設定)をクリックしてから、**[Refresh Preview]** (プレビューの更新)をクリックします。**[Next]** (次へ)をクリックして、最後の手順に進みます。
   - 手順4では、ファイルの各カラムを設定します。ウィザードには、ファイルの最初のデータ行に基づいて、カラムの型と長さを推定するアルゴリズムが組み込まれています。提示されたデータ記述(Talend Studioではスキーマと呼ばれます)はいつでも変更できます。このシナリオでは、それらをそのまま使用できます。

   これで、**California_clients**メタデータは完了です。

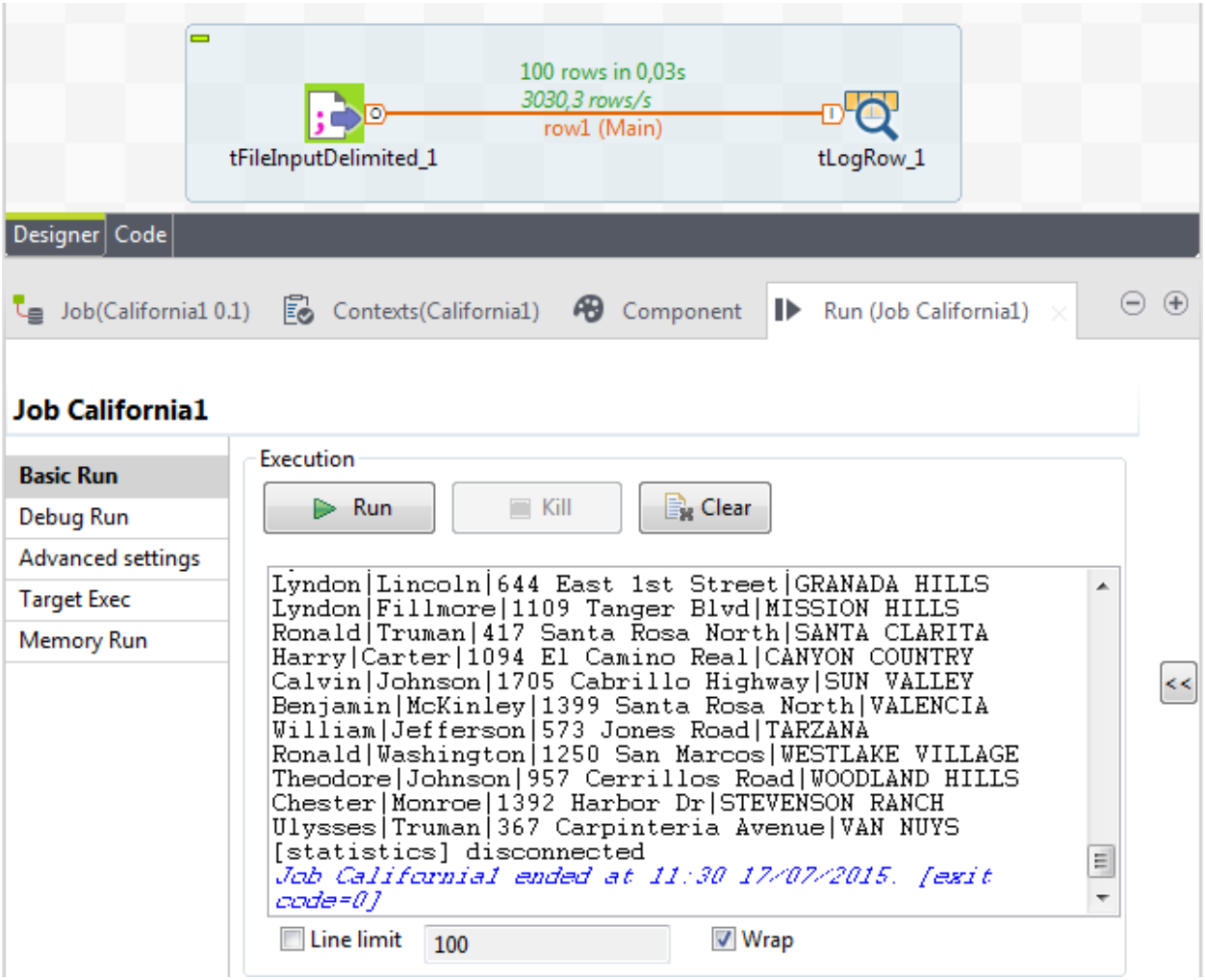
   これを入力コンポーネントで使用できます。先にデザインワークスペースにドロップした**tFileInputDelimited**を選択し、ウィンドウ下部に位置する**[Component]** (コンポーネント)ビューを選択します。
7. 縦に並んだタブの**[Basic settings]** (基本設定)を選択します。このタブには、コンポーネントを動作させるために必要なすべての技術的なプロパティが含まれています。これらのプロパティをそれぞれ設定するのではなく、定義したばかりのメタデータエントリを使用します。
8. リストで**[Property type]** (プロパティタイプ)として**[Repository]** (リポジトリ)を選択します。新しいフィールドが表示されるので、**[Repository]** (リポジトリ) の**[...]**ボタンをクリックし、リストから関連するメタデータエントリ**[California_clients]**を選択します。

   すべてのパラメーターが自動的に入力されたことを確認できます。



この段階では、単に入力ファイルから読み取ったデータを標準出力(StdOut)に送信することで、フローを終了します。

9. そのために、**tLogRow**コンポーネントを(**[Logs & Errors]** (ログ&エラー)グループから)追加します。両方のコンポーネントをリンクさせるため、入力コンポーネントを右クリックして、**[Row] (行)** > **[Main] (メイン)**を選択します。次に、出力コンポーネントの**tLogRow**をクリックします。
10. これで、このジョブを実行する準備ができました。実行するには、下部のパネルで**[Run]** (実行)タブを選択します。
11. **[Run]** (実行)ビューの縦に並んだタブの**[Advanced Settings]** (詳細設定)で**[Statistics]** (統計)チェックボックスをオンにして統計を有効にしてから、**[Basic Run]** (基本実行)タブで**[Run]** (実行)をクリックしてジョブを実行します。



入力ファイルの内容がコンソールに表示されます。

手順2: マッピングと変換

次に、ジョブを拡張してオンザフライ変換を組み込みます。これらの変換を実装するには、**tMap**コンポーネントをジョブに追加する必要があります。このコンポーネントは多機能で、次のことを処理できます。

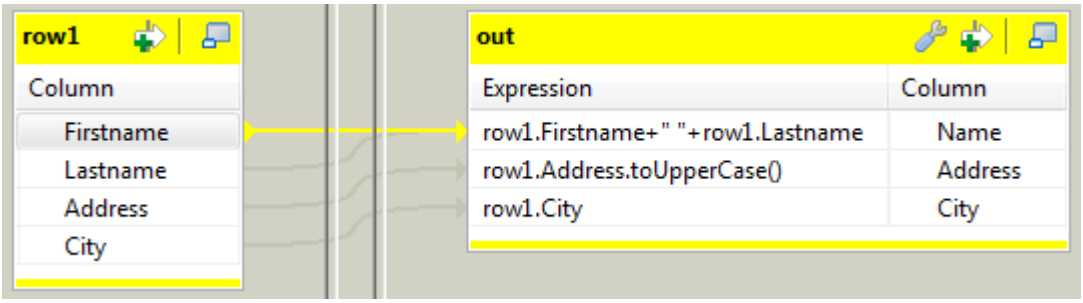
- 複数の入力および出力
- 参照の検索(単純、直積集合、前方一致、後方一致など)
- 結合(内部、外部)
- 変換
- リジェクト
- その他

手順

1. リンクを右クリックし、**[Delete]** (削除)オプションをクリックして、ジョブの2つのコンポーネントを結合するリンクを削除します。次に、以前行ったように、**[Processing]** (処理)コンポーネントグループの**tMap**を間に配置して、入力コンポーネントを**tMap**にリンクさせます。

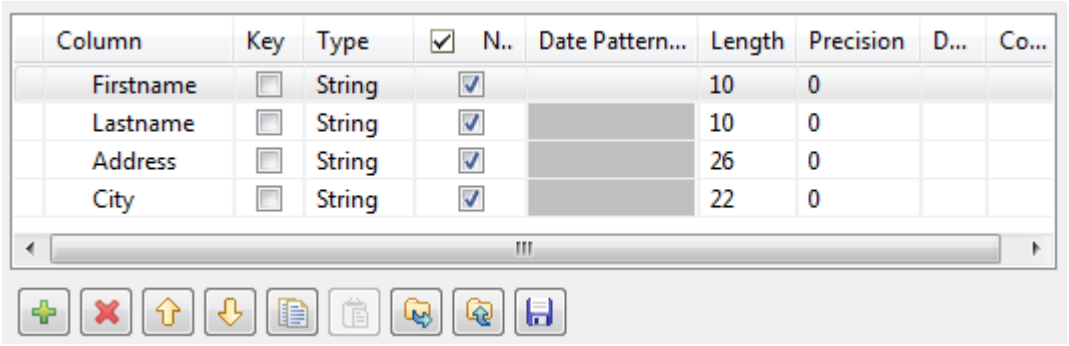
2. そして最後に、**tMap**を標準出力にリンクさせるので、**tMap**コンポーネントを右クリックして、**[Row] (行)** > **[*New Output* (Main)] (*新規出力* (メイン))**を選択して、**tLogRow**コンポーネントをクリックします。ダイアログボックスに`out1`と入力して、リンクを実装します。論理的には、ここでメッセージボックスが表示されますが(スキーマのバックプロパゲーションのため)、**[No]** (いいえ)をクリックして無視します。
3. 次に、**tMap**をダブルクリックして、そのインタフェースにアクセスします。

   左側で、入力ファイル(**row1**)のスキーマ(記述)を確認できます。右側の出力は現時点では空の状態です(**out1**)。
4. **[Firstname]** (名)カラムと**[Lastname]** (姓)カラムを、画面下部に示された**[Name]** (名前)カラムにドロップします。次に、**[Address]** (住所)カラムと**[City]** (都市)カラムを各行にドロップします。



5. 各カラムで次の変換を実行します。

   - **[Name]** (名前)カラムの式を`row1.Firstname + " " + row1.LastName`に変更します。これにより、**[Firstname]** (名)カラムの値と**[Lastname]** (姓)カラムの値が連結して1つのカラムに表示されるようになります。
   - **[Address]** (住所)カラムの式を`row1.Address.toUpperCase()`に変更して、住所テキストが大文字に変換されるようにします。
6. 次に、out1テーブルから[Lastname] (姓)カラムを削除し、残りのカラムの長さを増やします。それには、tMapエディターの下部にある**[Schema Editor]** (スキーマエディター)に移動し、次の手順に従います。



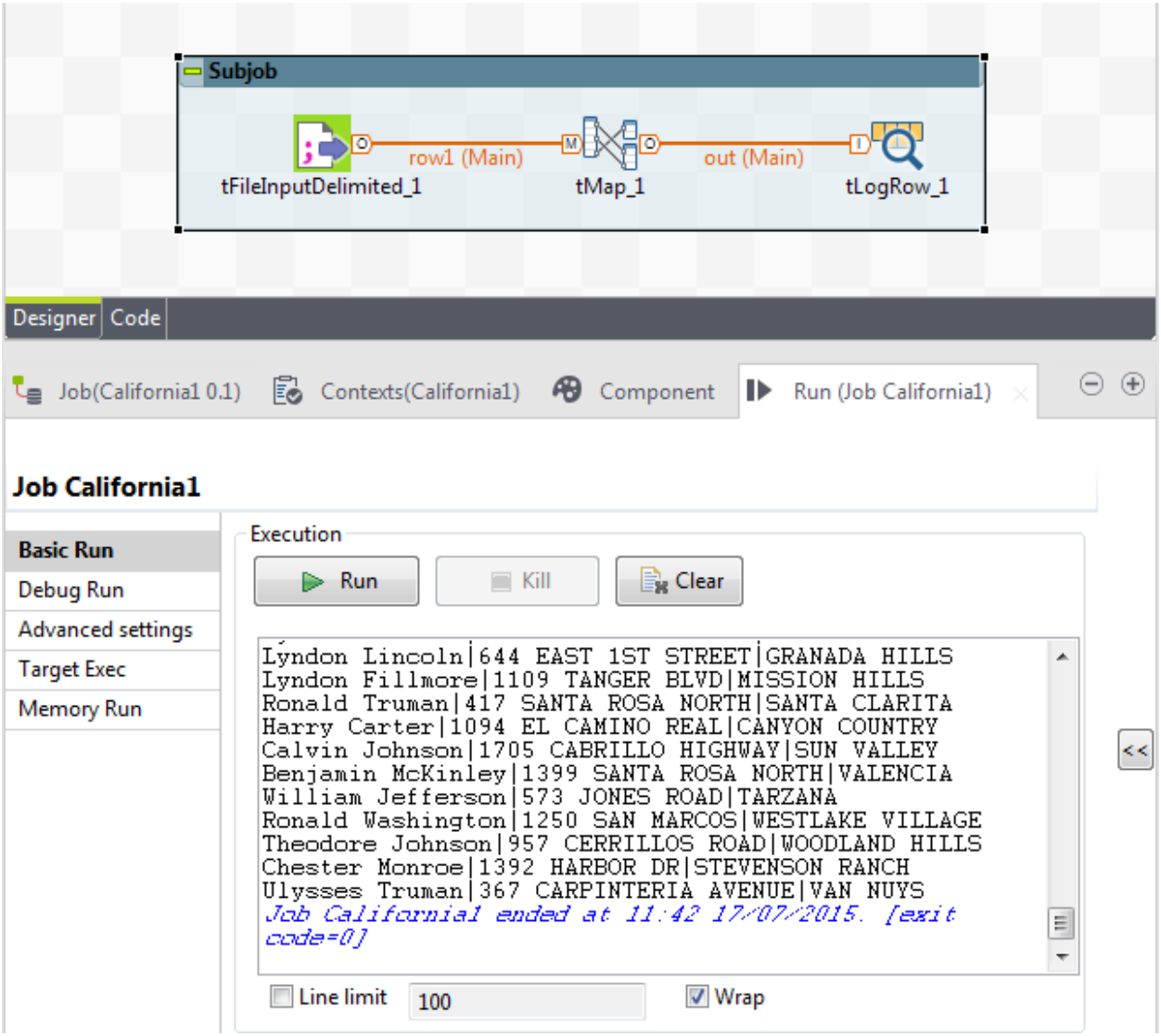
   1. スキーマから削除するカラムを選択し、十字アイコンをクリックします。
   2. 長さのサイズを増やす必要のあるカラムを選択します。
   3. [Length] (長さ)カラムに、長さサイズを入力します。この例では、残りの各カラムの長さを40に変更します。

注：

顧客の名と姓が連結されるため、フルネームのサイズに適合するように、名前カラムの長さを伸ばす必要があります。

[City] (都市)カラムでは、変換は行いません。

7. **[OK]**をクリックして変更を確認し、マップエディターインタフェースを閉じます。
8. この段階で(以前実行したように、**[Run]** (実行)ビューを介して)ジョブを実行すると、定義した変更が実装されていることがわかります。



住所は大文字で表示され、名と姓が同じカラムに一緒に表示されています。

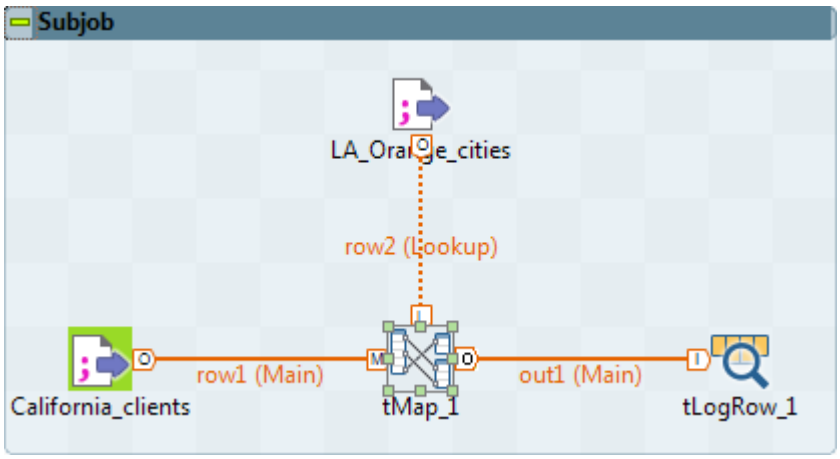
手順3: 参照ファイルの定義、再マッピング、内部結合モードの選択

手順

1. ウィザードを使用して`LosAngelesandOrangeCounties.txt`ファイルに対して実行したように、`California_clients`ファイルに対応するメタデータを定義します。

   ウィザードの手順1で、このメタデータエントリに`LA_Orange_cities`という名前を付けます。
2. この新たに作成したメタデータをデザインエリア上にドロップすると、このメタデータをポイントする読み取りコンポーネントが自動的に作成されます。

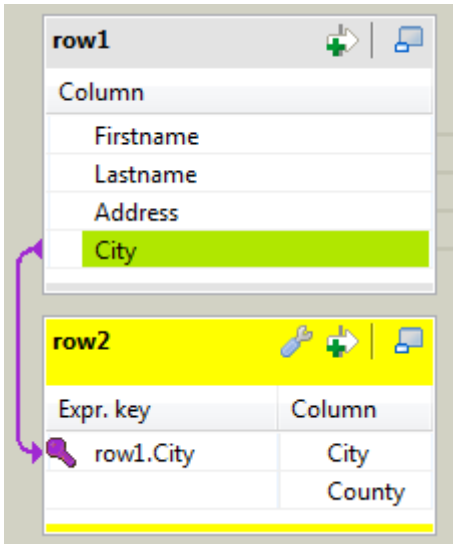
3. 次に、このコンポーネントを**tMap**コンポーネントにリンクさせます。



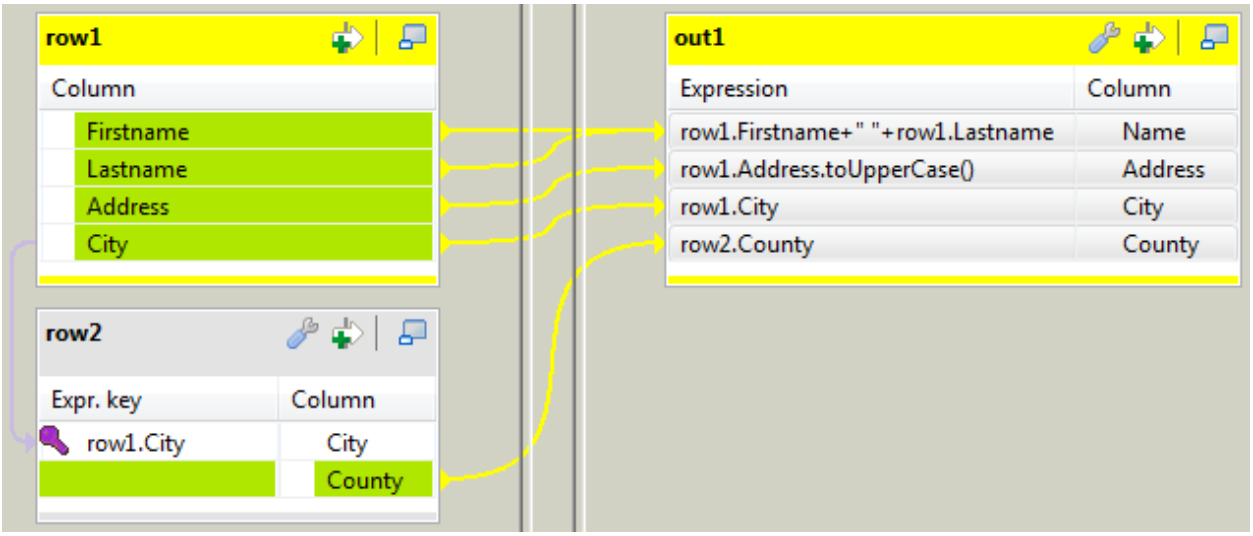
4. **tMap**コンポーネントを再度ダブルクリックして、このコンポーネントのインタフェースを開きます。LA郡とオレンジ郡のファイルに対応する参照入力テーブル(**row2**)が、ウィンドウ左側のメイン入力(**row1**)の下に表示されます。
5. 次に、メインフローと参照フローの間の結合を定義しましょう。

   このユースケースでは、**[City]** (都市)カラムが両方のファイルに存在しており、かつそのデータが完全に一致しているため、結合の定義はかなり基本的なものになります。ただし、このような場合でなくとも、このレベルで直接演算を実行して、データ間のリンクを確立できます(埋め込み、大文字小文字の変更など)。

   結合を実装するには、最初の入力テーブルの**[City]** (都市)カラムを、参照テーブルの**[City]** (都市)カラムにドロップします。紫色のリンクが表示され、この結合が実体化されます。

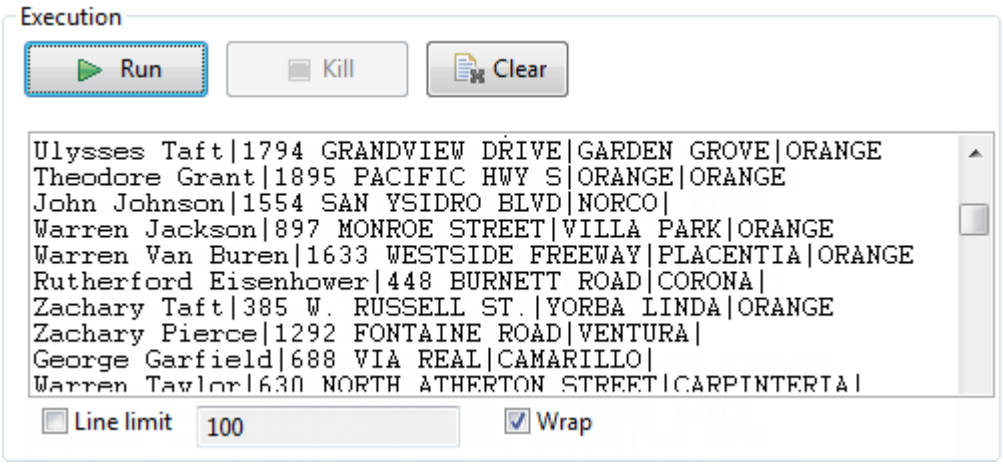


   これで、参照テーブルの**[County]** (郡)カラムを出力テーブル(**out1**)で使用できるようになりました。



6. 最後に、**[OK]**をクリックして変更を確定し、新しいジョブを実行します。

   次のような出力がコンソールに表示されます。



   ご覧のとおり、最後のカラムには、ロサンゼルス郡とオレンジ郡の都市である場合のみ、値が存在しています。その他の行については、このカラムは空になっています。これは、デフォルトで、**tMap**が左側外部結合モードになっているからです。データをフィルター処理して**tMap**によって一致が見つかった行のみ表示する場合は、**tMap**を再度開き、**[tMap settings]** (tMap設定)ボタンをクリックして、参照テーブル(**row2**)の**[Join Model]** (結合モデル)リストで、**[Inner Join]** (内部結合)を選択します。

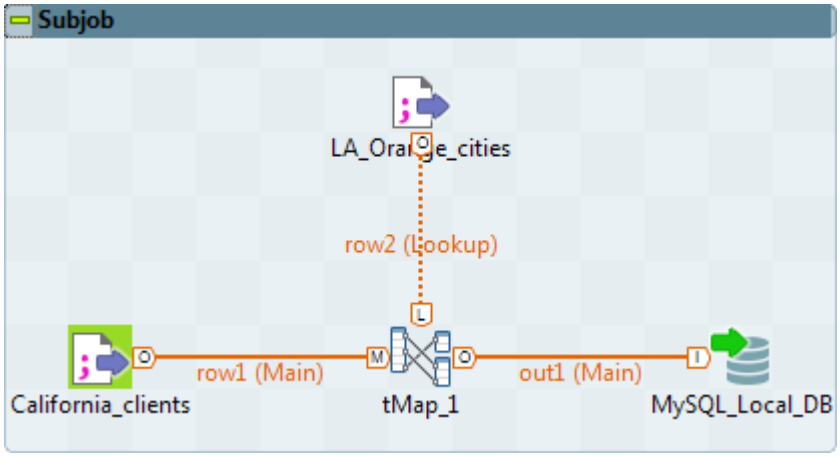
手順4: MySQLテーブルへの出力

ここまで、ジョブは完璧に動作しています。仕上げとして、出力フローをMySQLテーブルに流してみましょう。

手順

1. そのためには、まず、MySQLデータベースへの接続を記述するメタデータを作成します。リポジトリで**[Metadata] (メタデータ)** > **[MySQL]**ノードを展開し、**[DemoMySQL]**をダブルクリックします (デモプロジェクトが適切にインポートされている場合)。メタデータウィザードが開きます。

2. ウィザードの手順2で、関連する接続パラメーターを入力します。**[Check]** (チェック)ボタンをクリックして、この接続の有効性を確認します。最後に、変更を確認して、**[Finish]** (終了)をクリックします。
3. **tMysqlOutput**コンポーネントを自動的に作成するために、**Ctrl**キーを押したまま、このメタデータをデザインワークスペースの右側にドロップします。
4. **tLogRow**コンポーネントをジョブから削除します。
5. **tMap**からの**out1**出力フローを、新しいコンポーネントの**tMysqlOutput**に再接続します。



6. このコンポーネントの**[Basic Settings]** (基本設定)タブで、以下のことを行います。
   a) `[Table]` (テーブル)フィールドに**LA_Orange_Clients**と入力し、オンザフライで作成されるターゲットテーブルに名前を付けます。
   b) **[Action on table]** (テーブル操作)フィールドで**[Drop table if exists and create]** (テーブルが存在する場合、削除してから作成)オプションを選択します。
   c) 必要に応じて、**[Edit Schema]** (スキーマの編集)をクリックし、**[Reset DB type]** (DBタイプのリセット)ボタン(ツールバーのDBボタン)をクリックして、DBタイプを自動的に入力します。
7. ジョブを再度実行します。

タスクの結果

ターゲットテーブルが、1秒未満で、自動的に作成され、データが入力されます。

このシナリオでは、**[Palette]** (パレット)で使用可能な、異なるカテゴリ(データベース、Webサービス、FTPなど)に従ってグループ化された数百のコンポーネントのうち4つのコンポーネントのみを使用しました。

さらに、コミュニティによって作成された多くのコンポーネントもコミュニティサイト(talendforge.org)で入手可能です。

出力ストリームフィーチャーの使用

次のユースケースでは、出力ストリームフィーチャーを使って、多数のコンポーネントでの出力パフォーマンスを大幅に向上させる方法を示します。

このシナリオでは、顧客情報が格納された定義済みのcsvファイルをデータベーステーブルにロードします。次に、**tMap**を使ってロードしたデータを選択し、出力ストリームフィーチャーを使ってそのデータをローカルファイルとコンソールに出力します。

Use Output Stream (出力ストリームの使用)フィーチャーの基本概念については、「**出力ストリームの使用フィーチャーを使用する方法**」(https://help.talend.comのTalend Studioユーザーガイド)を参照して下さい。

入力データ

入力ファイル(このデータがデータベーステーブルにロード)には、さまざまな視点からの顧客情報が含まれています。

このファイル構造(Talend Studioでは通常**スキーマ**と呼ばれている)には、次のカラムが含まれています。

- id (Integer型)
- CustomerName (String型)
- CustomerAge (Integer型)
- CustomerAddress (String型)
- CustomerCity (String型)
- RegisterTime (Date型)

出力データ

tMapコンポーネントを使って、入力データから**id**、**CustomerName**、**CustomerAge**の各カラムを選択します。次に、出力ストリームフィーチャーを使って、選択したデータを出力します。

予想される出力データは、次の構造になります。

- id (Integer型)
- CustomerName (String型)
- CustomerAge (Integer型)

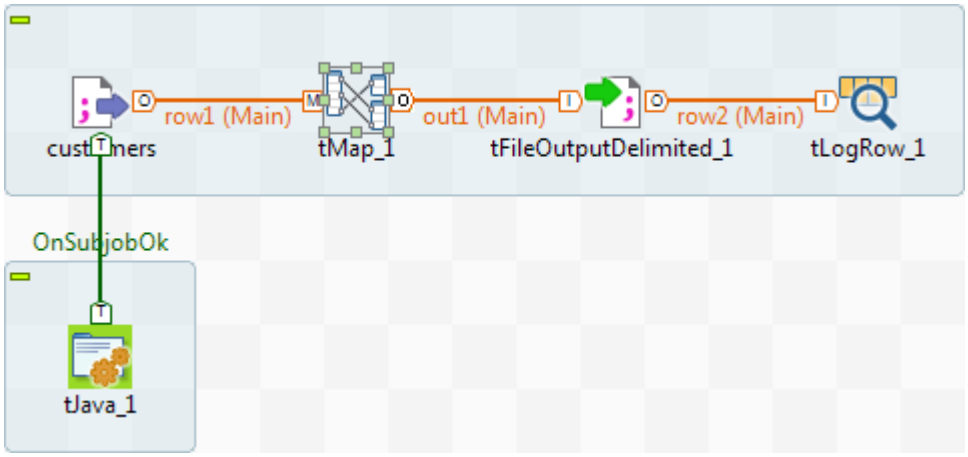
上記の3つのカラムは、入力データの該当するカラムから取得されます。

シナリオをジョブに変換する

このシナリオを実装するには、ジョブに関する次の4つの手順が必要です。

1. ジョブの作成、入力データのスキーマの定義、定義済みスキーマを基にした入力ファイルの読み取り。
2. 出力ストリームフィーチャーを有効にするコマンドの設定。
3. **tMap**コンポーネントを使ったデータのマッピング。
4. 選択したデータストリームの出力。

ジョブ全体は、次の画像に示されているような流れになります。ジョブの設計の詳細は、以降のセクションを参照して下さい。

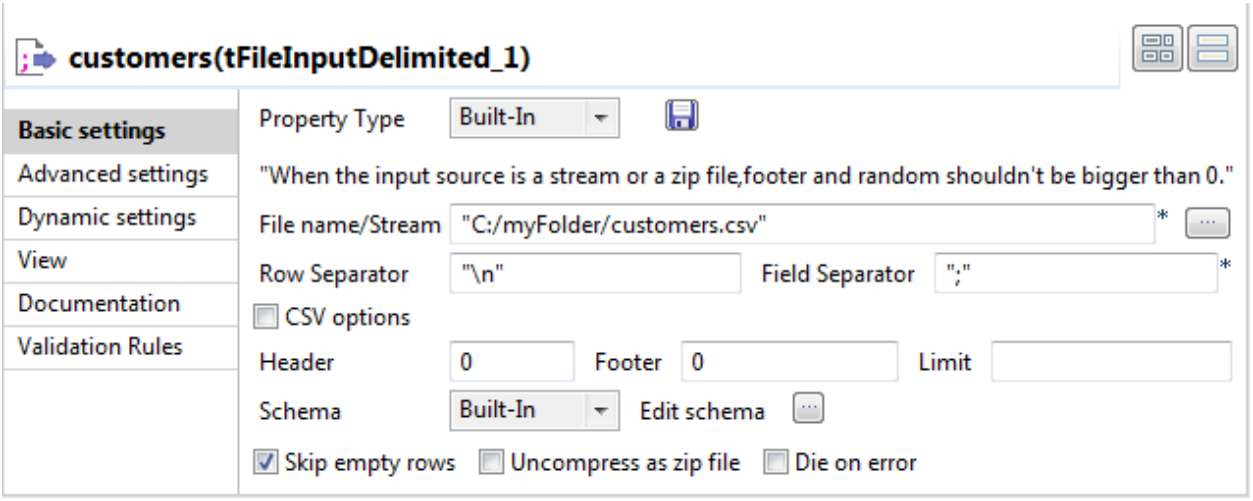


手順1: ローカルファイルから入力データを読み取る

tFileInputDelimitedコンポーネントを使って、入力データとして`customers.csv`ファイルを読み取ります。このコンポーネントは、**[Palette]** (パレット)の**[File/Input]** (ファイル/入力)グループにあります。

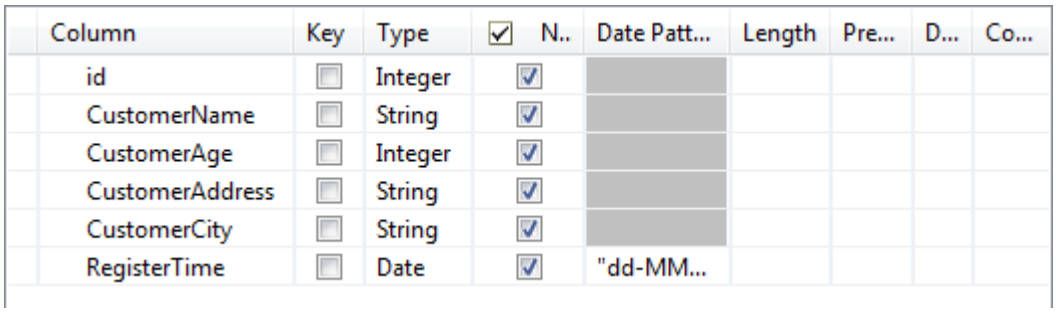
手順

1. **tFileInputDelimited**コンポーネントをデザインワークスペースにドロップしてダブルクリックし、**[Basic settings]** (基本設定)ビューを開いて、プロパティを設定します。



2. **[File name/Stream]** (ファイル名/ストリーム)の横の[...]ボタンをクリックし、入力データファイルのパスを参照して選択します。入力データファイルのパスは手動で入力することもできます。

3. **[Edit schema]** (スキーマの編集)をクリックしてダイアログボックスを開き、入力ファイルのファイル構造を設定します。
4. [+]ボタンをクリックしてカラムを6つ追加し、次の画像のように、**[Type]** (タイプ)とカラム名を設定します。



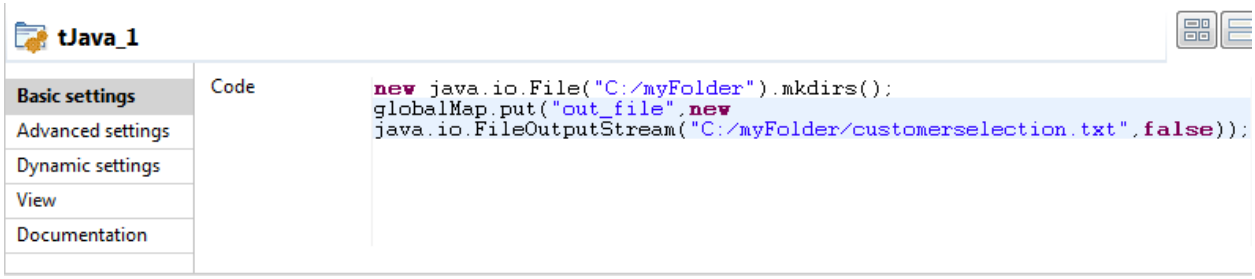
5. **[OK]**をクリックして、ダイアログボックスを閉じます。

手順2: 出力ストリームフィーチャーを有効にするコマンドを設定する

ここでは、**tJava**を使って、出力ファイルと、そのファイルを含むディレクトリを作成するコマンドを設定します。

手順

1. **tJava**コンポーネントをデザインワークスペースにドロップしてダブルクリックし、**[Basic settings]** (基本設定)ビューを開いて、プロパティを設定します。

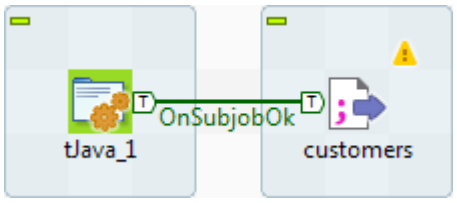


2. **[Code]** (コード)エリアに次のコマンドを入力します。

```
new java.io.File("C:/myFolder").mkdirs();
globalMap.put("out_file",new java.io.FileOutputStream("C:/myFolder/cust
omerselection.txt",false));
```

> **注：**
>
> ここで入力したコマンドにより、出力ファイル`C:/myFolder`を保存するための新規ディレクトリ`customerselection.txt`が作成されます。実際の状況に応じて、コマンドをカスタマイズできます。

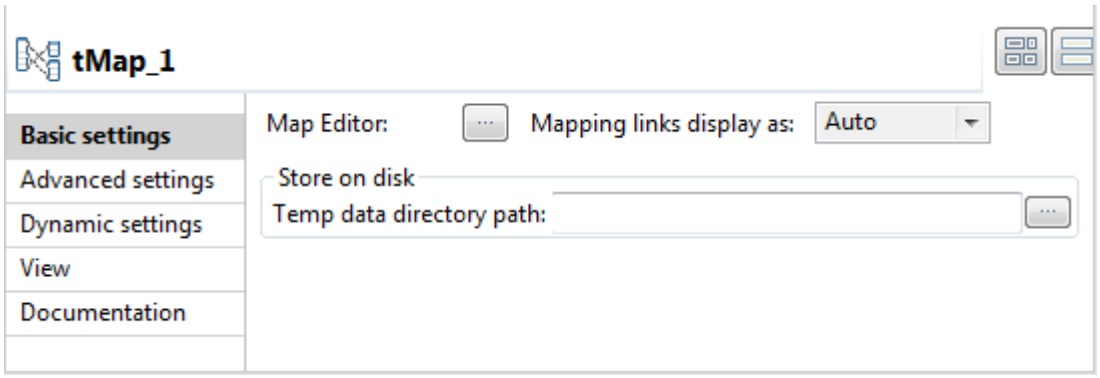
3. **[Trigger] (トリガー)** > **[On Subjob Ok] (サブジョブがOKの場合)**接続を使って、**tJava**と**tFileInputDelimited**を接続します。これにより、**tFileInputDelimited**で開始するサブジョブの実行が正常に行われたときに、**tJava**がトリガーされます。



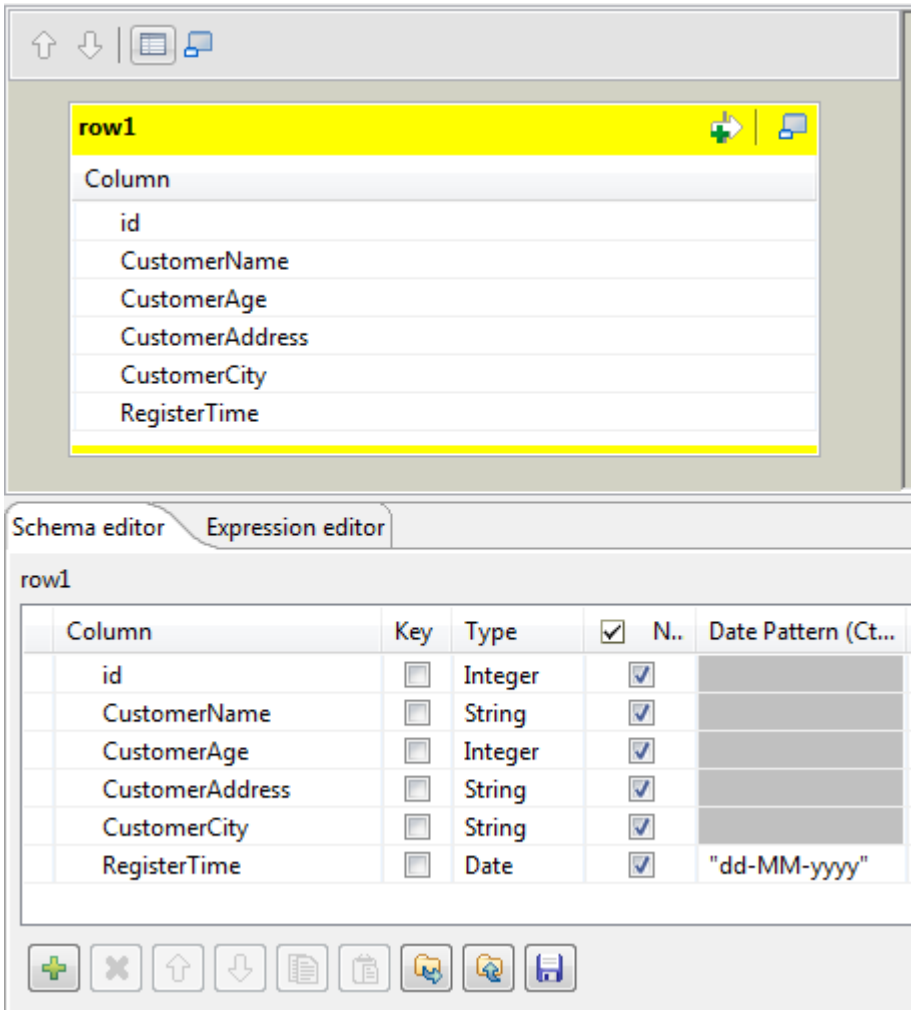
手順3: tMapコンポーネントを使ってデータのマッピングを行う

手順

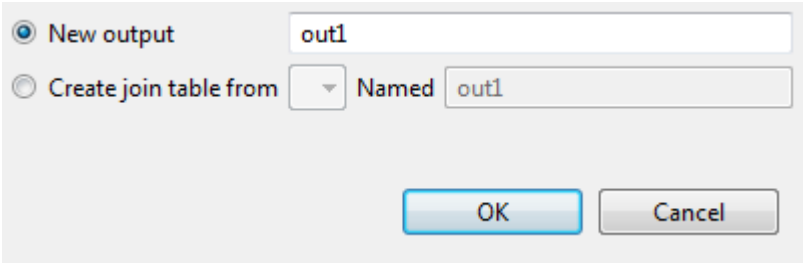
1. **tMap**コンポーネントをデザインワークスペースにドロップしてダブルクリックし、**[Basic settings]** (基本設定)ビューを開いて、プロパティを設定します。



2. **[Map Editor]** (マップエディター)の横にある[...]ボタンをクリックし、ダイアログボックスを開いて、マッピングを設定します。
3. 左側の[+]ボタンをクリックし、入力側データのスキーマのためにカラムを6つ追加します。これらのカラムは次の画像と同じでなければなりません。



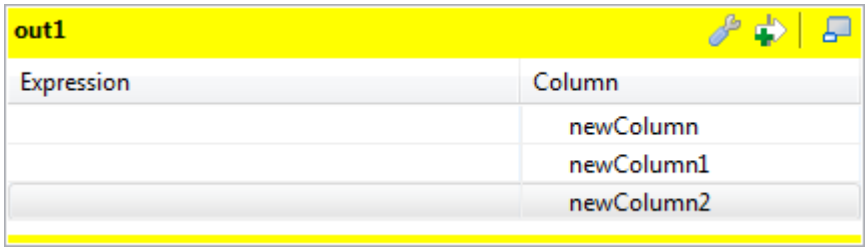
4. 右側の[+]ボタンをクリックし、出力側データフローのスキーマを追加します。



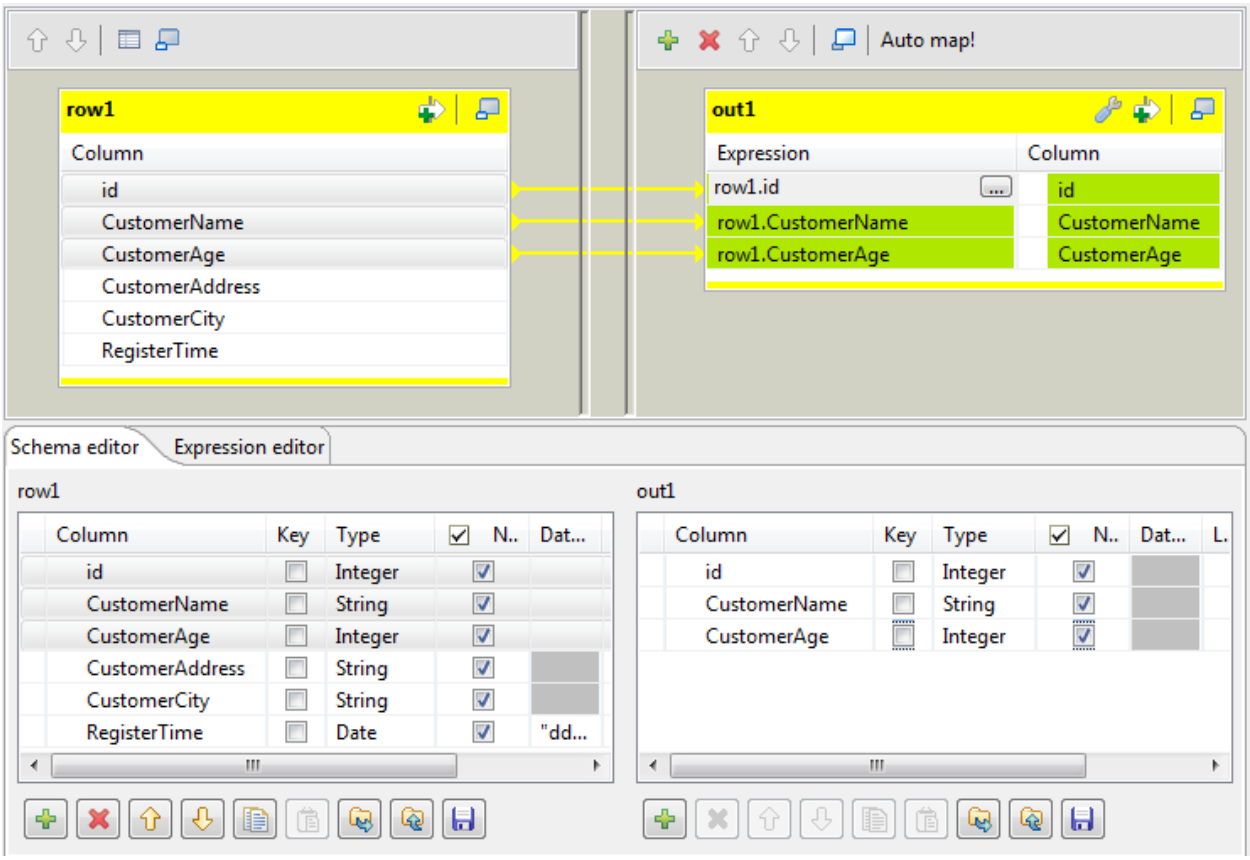
5. **[New output]** (新規出力)を選択し、**[OK]**をクリックして、出力スキーマを保存します。

   現在のところ、出力スキーマはまだ空の状態です。

6. **out1**テーブルの下にある[+]ボタンをクリックし、出力データのカラムを3つ追加します。



7. **id**、**CustomerName**、**CustomerAge**の各カラムを右側のそれぞれの行にドロップします。



8. **[OK]**をクリックして、設定を保存します。

手順4: 選択したデータストリームを出力する

手順

1. **tFileOutputDelimited**コンポーネントをデザインワークスペースにドロップしてダブルクリックし、**[Basic settings]** (基本設定)ビューを開いて、コンポーネントのプロパティを設定します。
2. **[Use Output Stream]** (出力ストリームの使用)チェックボックスをオンにし、**[Output Stream]** (出力ストリーム)フィールドを有効にして、**[Output Stream]** (出力ストリーム)フィールドに次のコマンドを入力します。

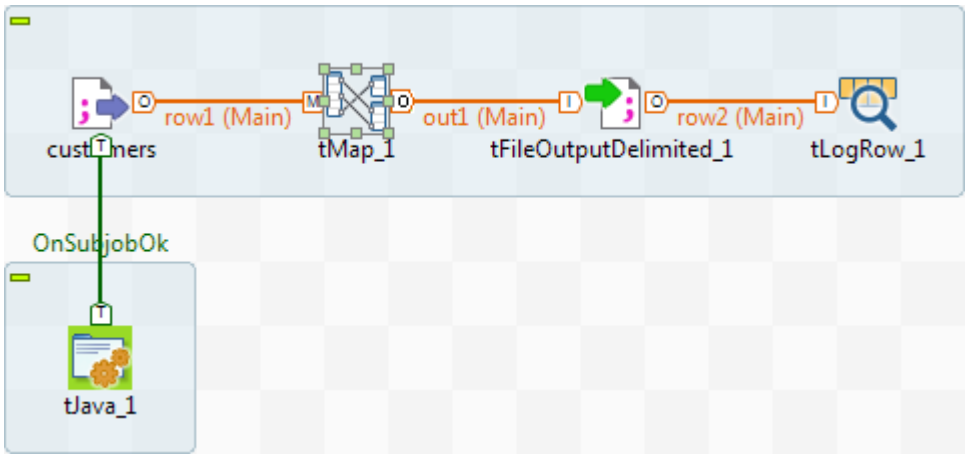
```
(java.io.OutputStream)globalMap.get("out_file")
```

> **注 :**
>
> **[Output Stream]** (出力ストリーム)フィールドのコマンドは、**Ctrl+Space**を押して、リストから組み込みコマンドを選択するか、実際の状況に従って、フィールドにコマンドを手動で入力します。このシナリオでは、**[Output Stream]** (出力ストリーム)フィールドで使用するコマンドにより、`java.io.OutputStream`クラスが呼び出され、絞り込んだデータストリームを、**tJava**の**[Code]** (コード)エリアで定義するローカルファイルに出力します。

3. **[Row] (行)** > **[Main] (メイン)**接続を使って**tFileInputDelimited**を**tMap**に接続し、**tMap**の**[Map Editor]** (マップエディター)で定義する**[Row] (行)** > **[out1]**接続を使って**tMap**を**tFileOutputDelimited**に接続します。

4. **[Sync columns]** (カラムの同期)タブをクリックし、先行のコンポーネントで定義されているスキーマを取得します。
5. **tLogRow**コンポーネントをデザインワークスペースにドロップしてダブルクリックし、**[Basic settings]** (基本設定)ビューを開きます。
6. **[Mode]** (モード)エリアの**[Table]** (テーブル)ラジオボタンを選択します。
7. **[Row] (行)** > **[Main] (メイン)**接続を使って**tFileOutputDelimited**を**tLogRow**に接続します。
8. **[Sync columns]** (カラムの同期)タブをクリックし、先行のコンポーネントで定義されているスキーマを取得します。

   これでこのジョブを実行する準備ができました。



9. **Ctrl+S**を押してジョブを保存し、**F6**を押して実行します。

   選択したデータの内容がコンソールに表示されます。

```
Starting job OutputStream at 17:31 19/10/2011.

[statistics] connecting to socket on port 4059
[statistics] connected
.-----+------------+-----------.
|           tLogRow_1           |
|=----+------------+-----------=|
|id   |CustomerName|CustomerAge|
|=----+------------+-----------=|
|10001|Warren      |67         |
|10002|Woodrow     |68         |
|10003|Grover      |77         |
|10004|Abraham     |74         |
|10005|Chester     |78         |
|10006|Calvin      |63         |
|10007|Zachary     |53         |
|10008|Chester     |36         |
|10009|Chester     |60         |
|10010|Woodrow     |57         |
'-----+------------+-----------'
[statistics] disconnected
Job OutputStream ended at 17:31 19/10/2011. [exit code=0]
```

   選択したデータは、指定したローカルファイル`customerselection.txt`にも出力されます。

customerselection.txt

```
id;CustomerName;CustomerAge
10001;Warren;67
10002;Woodrow;68
10003;Grover;77
10004;Abraham;74
10005;Chester;78
10006;Calvin;63
10007;Zachary;53
10008;Chester;36
10009;Chester;60
10010;Woodrow;57
```

暗黙的なコンテキストのロードフィーチャーの使用

コンテキスト変数に基づくジョブのパラメーター化により、異なるコンテキストまたは環境でのジョブのオーケストレーションと実行に対応できます。コンテキスト変数の値は、作成中に定義する方法と、ジョブの実行中に明示的または暗黙的にコンテキストパラメーターを動的にロードする方法とがあります。

以下のユースケースでは、Talend Studioの暗黙的なコンテキストのロードフィーチャーを使用して、ジョブの実行中に動的にコンテキストパラメーターをロードする方法を示します。

このユースケースのジョブで使用するコンポーネントは2つだけです。テスト用と本番環境用の2つのMySQLデータベースに格納されている従業員データを読み込みます。この2つのデータベースにアクセスするための接続パラメーターは、別のMySQLデータベースにあります。ジョブの実行時、これらの接続パラメーターが動的にロードされ、2つのデータベースへの接続が行われます。

ジョブを作成してコンテキスト変数を定義する

始める前に

前述した`testing`と`production`の2つのデータベースにアクセスするための接続パラメーターを格納する`db_testing`と`db_production`という名前の2つのテーブルを、`db_connections`という名前のMySQLデータベースに作成します。 各テーブルは、VARCHAR型の2つのカラム`key`と`value`のみの構成にします。以下、データベーステーブルのコンテンツ例を示します。

db_testing:

key	**value**
host	localhost
port	3306
username	root
password	talend
database	testing

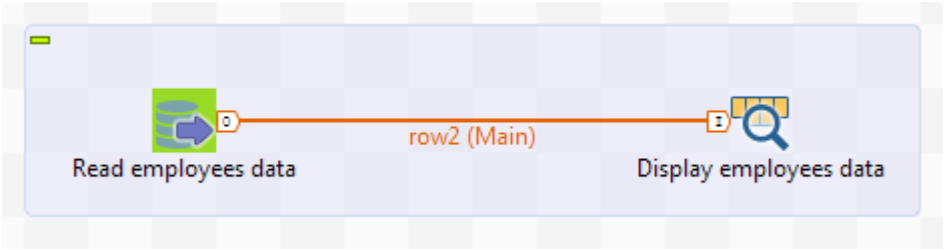
db_production:

key	**value**
host	localhost
port	3306
username	root
password	talend
database	production

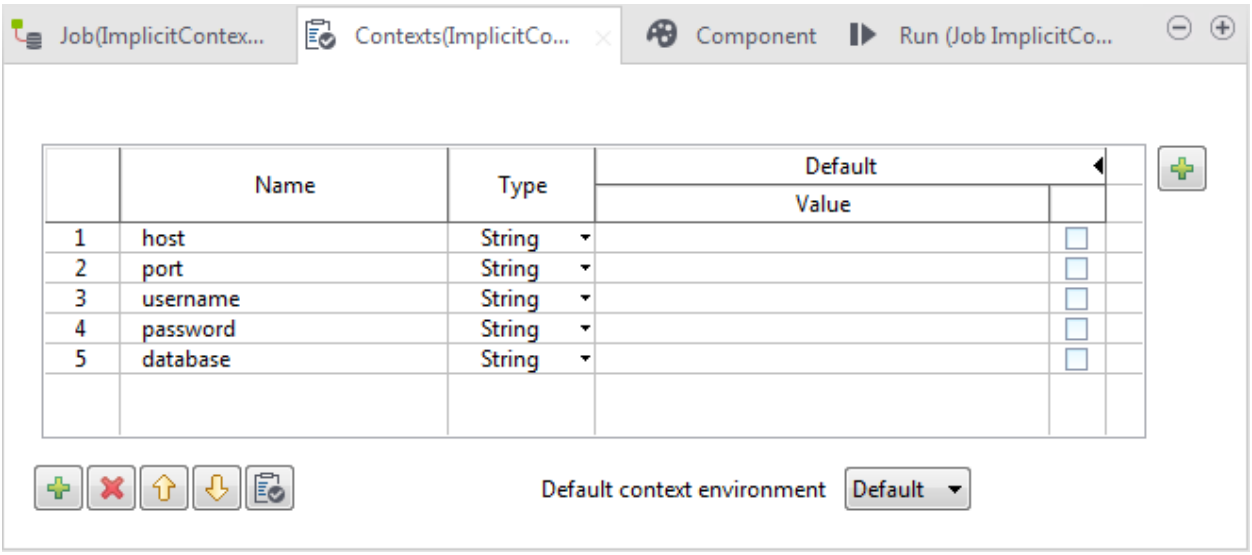
これらのデータベーステーブルは、**tFixedFlowInput**コンポーネントと**tMysqlOutput**コンポーネントを含む別のTalendジョブを使って作成できます。

手順

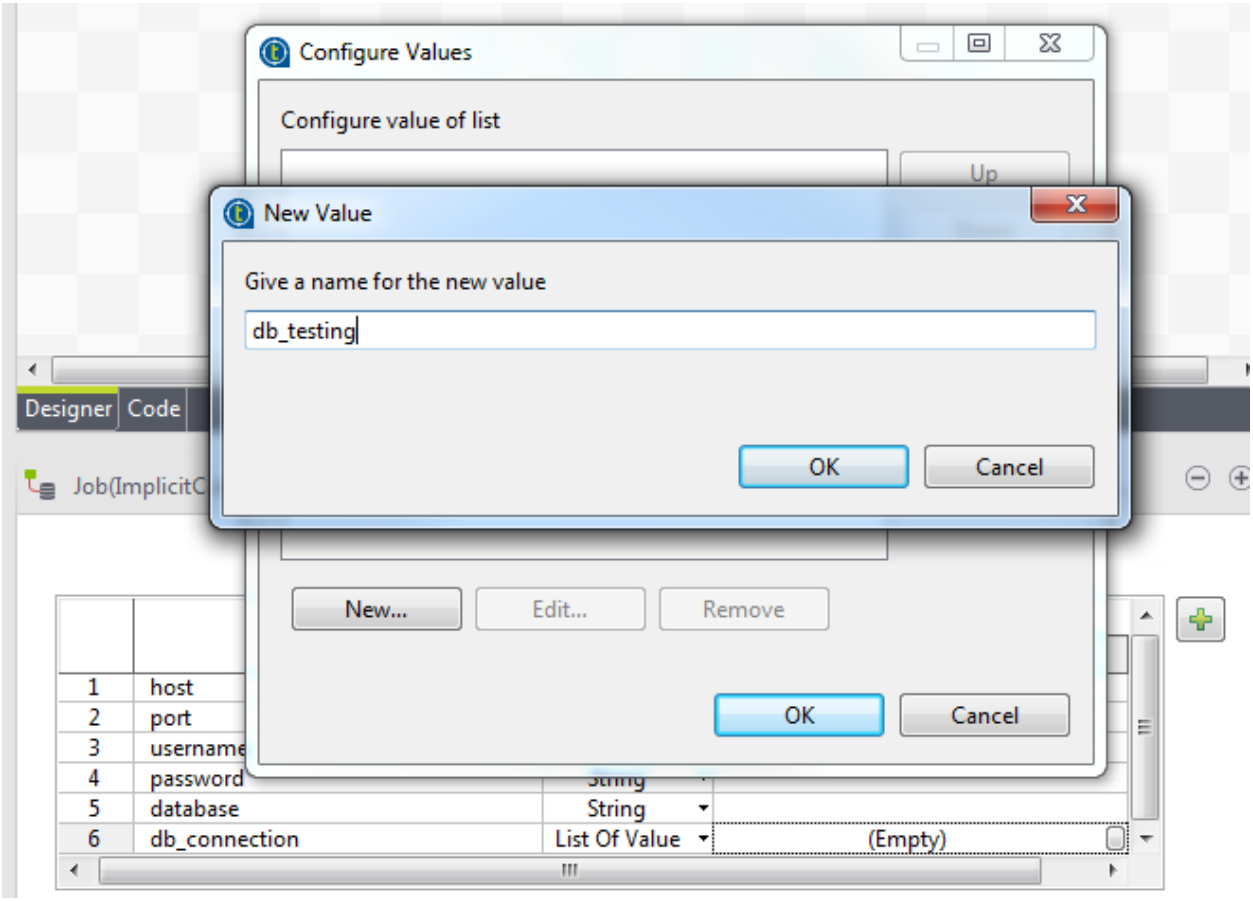
1. ジョブを作成し、**tMysqlInput**コンポーネントと**tLogRow**コンポーネントをデザインワークスペースに追加し、これらを**[Row] (行)** > **[Main] (メイン)**接続を使ってリンクさせます。



2. ジョブの**[Contexts] (コンテキスト)**ビューを選択し、ビューの下部にある**[+]**ボタンをクリックして、テーブルに5つの行を追加し、**host**、`port`、、`username`、`password`、`database`の変数を定義します。これらはすべて`String`型で、ジョブの実行中に動的にロードされます。



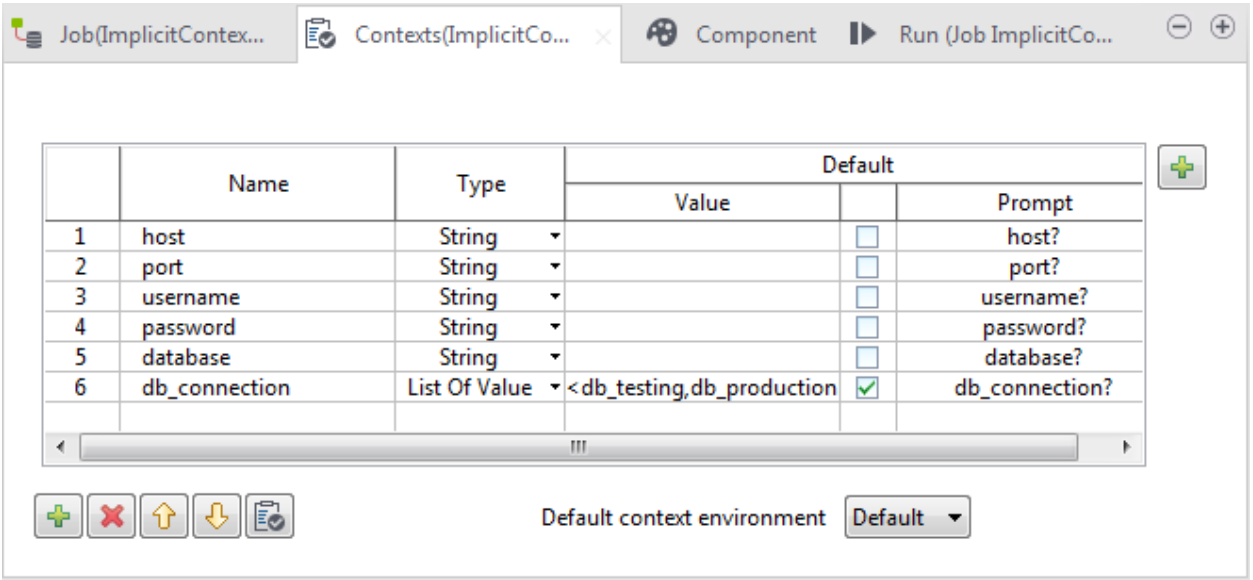
3. ここでもう一つ、**[List Of Value]** (値のリスト) 型の`db_connection`という変数を作成します。
4. 新しく作成した変数の**[Value]** (値)フィールド内をクリックして、表示されるボタンをクリックし、**[Configure Values]** (値の設定)ダイアグボックスを開き、**[New...]** (新規...)をクリックして**[New Value]** (新しい値)ダイアログボックスを開きます。データベース接続情報を格納するデータベーステーブルの1つの名前を入力し、**[OK]**をクリックします。



5. もう一度**[New...]** (新規...)をクリックしたら、データベース接続情報を格納するもう一つのテーブルを定義します。 定義後、**[OK]**をクリックして、**[Configure Values]** (値の設定)ダイアログボックスを閉じます。

   これで、変数`db_connection`に`db_testing`と`db_production`の値リストが設定されました。これらは、接続パラメーターをロードするデータベーステーブルです。

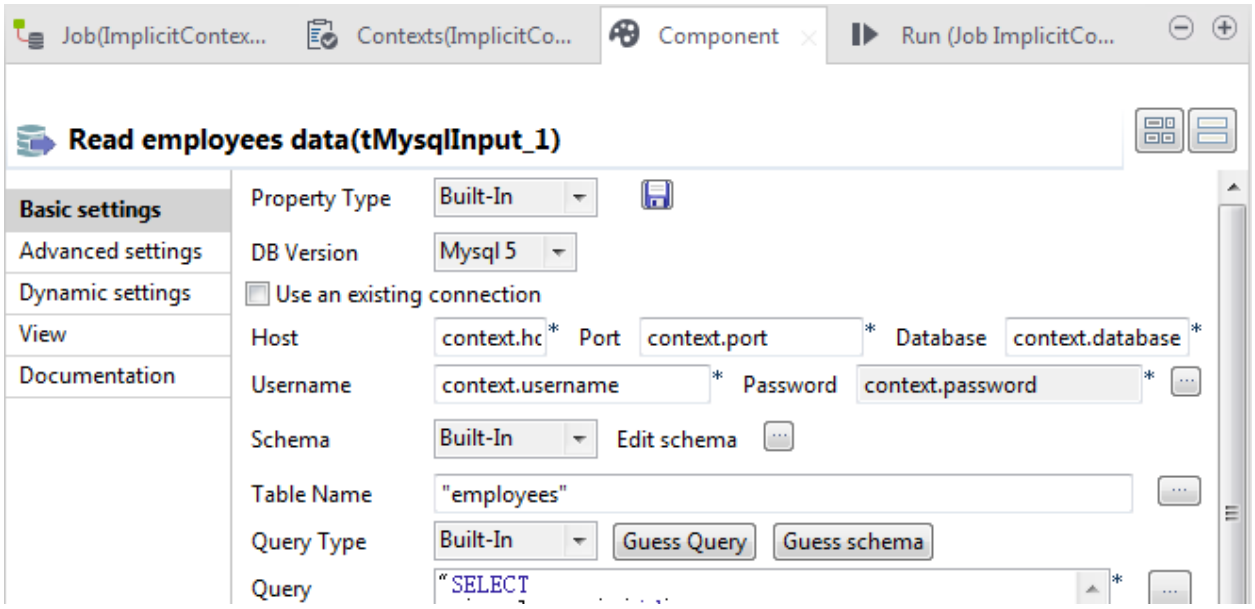
6. `db_connection`変数の**[Value] (値)** フィールドの横の**[Prompt] (プロンプト)**チェックボックスをオンにして、**[Prompt] (プロンプト)** フィールドを表示し、実行時に表示されるプロンプトメッセージを入力します。



コンポーネントを設定する

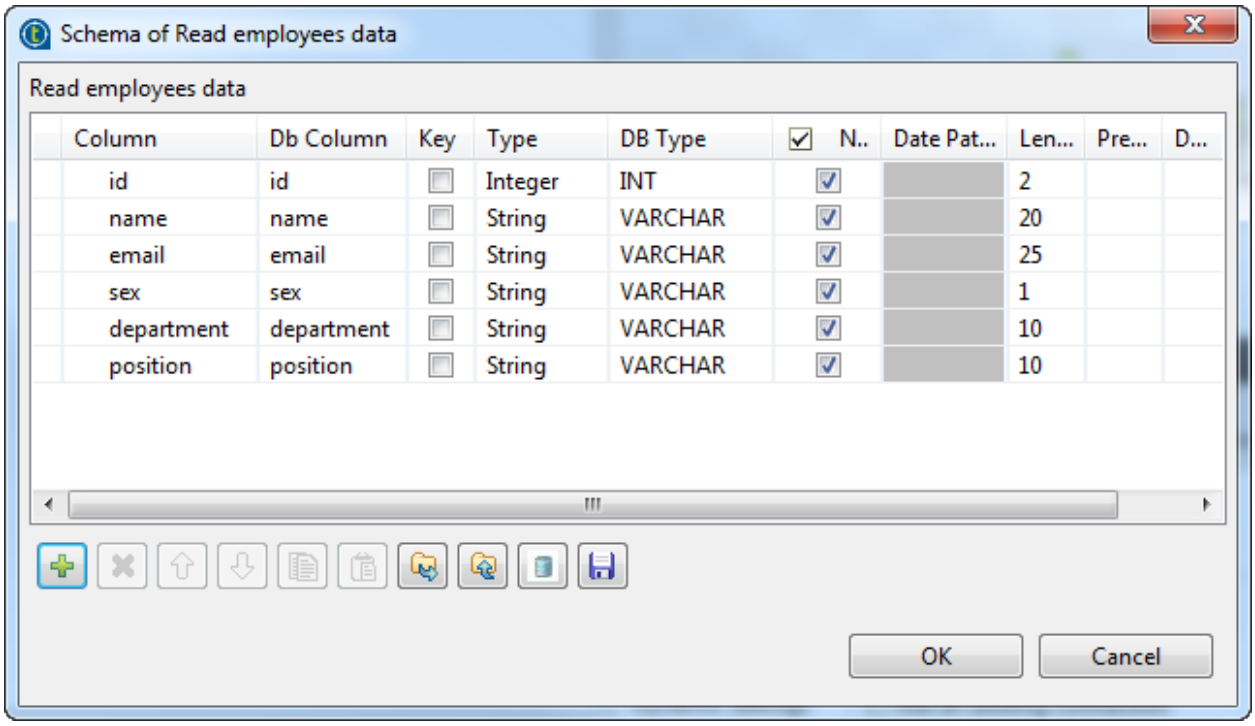
手順

1. **tMysqlInput**コンポーネントをダブルクリックして、**[Basic settings]** (基本設定)ビューを開きます。
2. **[Host]** (ホスト)、**[Port]** (ポート)、**[Database]** (データベース)、**[Username]** (ユーザー名)、**[Password]** (パスワード)、**[Table Name]** (テーブル名) の各フィールドに、**[Contexts]** (コンテキスト)タブビューで定義した関連する変数を入力します。この例ではそれぞれ`context.host`、`context.port`、`context.database`、`context.username`、`context.password`になります。



3. **[Table Name]** (テーブル名)フィールドに`employees`と入力します。これは、両方のデータベースにある従業員情報のテーブル名です。
4. 次に、**[Schema]** (スキーマ)情報を入力します。**リポジトリ**にスキーマを格納している場合は、**[Repository]** (リポジトリ)を選択して、リストから関連項目を選択することによって、スキーマを取得します。

   この例では、両方のデータベーステーブルのスキーマは、6つのカラムである`id` (INT型、2桁)、`name` (VARCHAR型、20文字)、`email` (VARCHAR型、25文字)、`sex` (VARCHAR型、1文字)、 `department` (VARCHAR型、10文字)、`position` (VARCHAR型、10文字)で構成されています。



5. **[Guess Query]** (クエリの推測)をクリックし、全テーブルカラムを取得します。これらは、**tLogRow**コンポーネント経由で**[Run]** (実行)タブに表示されます。
6. **tLogRow**コンポーネントの**[Basic settings]** (基本設定)ビューで、**[Table]** (テーブル)オプションを選択し、データレコードをテーブル形式で表示します。

暗黙的なコンテキストのロードフィーチャーを設定する

暗黙的なコンテキストのロードフィーチャーの設定は、プロジェクト内のすべてのジョブで使用できるようにプロジェクト設定で行うこともできますし、特定のジョブのみに適用されるようにジョブビューで行うこともできます。

次の例では、特定ジョブの**[Job]** (ジョブ)ビューで、暗黙的なコンテキストのロードフィーチャーを設定する方法を示します。フィーチャーを複数のジョブで再利用できるように設定するには、メニューバーから**[File] (ファイル)** > **[Edit Project properties] (プロジェクトプロパティの編集)**を選択して、**[Project Settings]** (プロジェクト設定)ダイアグボックスを開き、**[Job Settings] (ジョブ設定)** > **[Implicit context load] (暗黙的なコンテキストのロード)**の順に選択して、**[Implicit tContextLoad]** (暗黙的なtContextLoad)チェックボックスをオンにし、以下の手順2から6に従って、パラメーターを設定します。次に、**[Job]** (ジョブ)ビューで**[Use Project Settings]** (プロジェクト設定の使用)チェックボックスをオンにして、ジョブに設定を適用します。

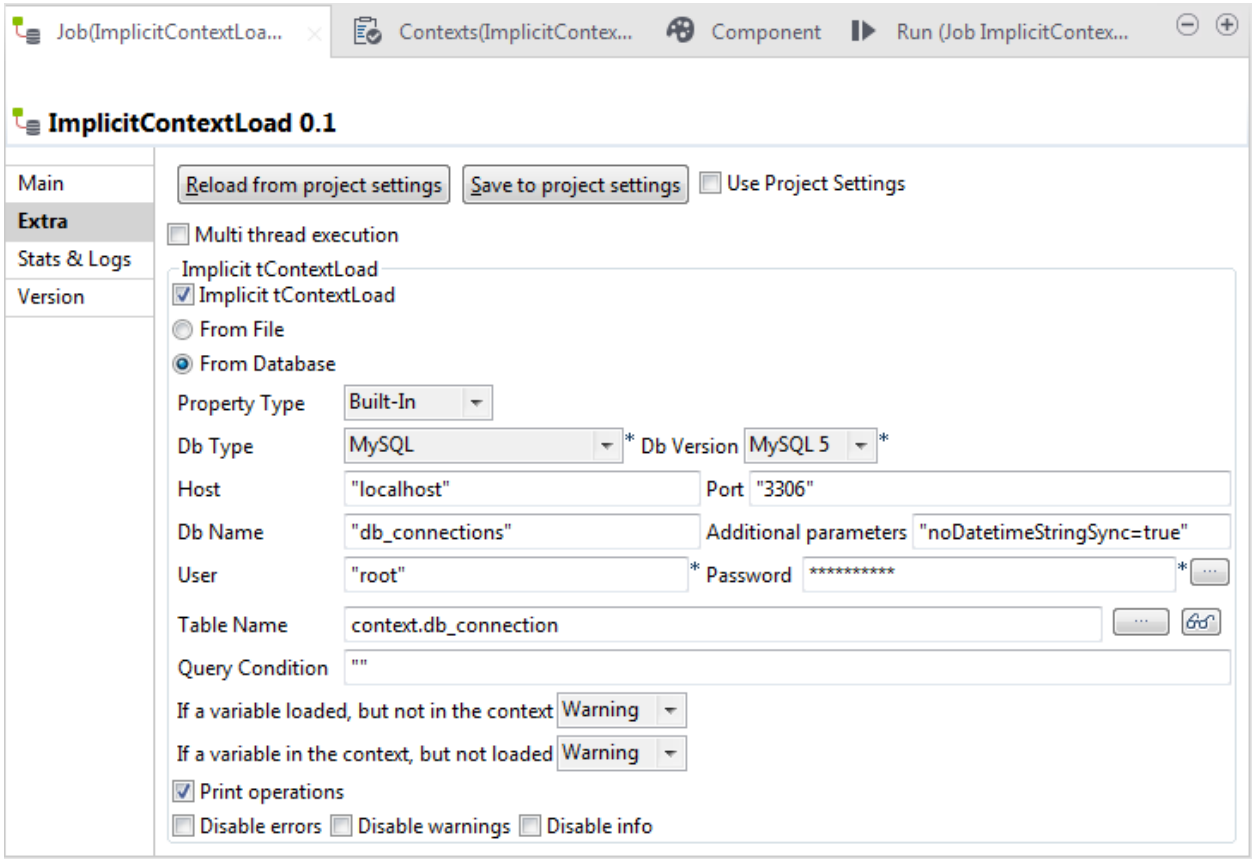
手順

1. **[Job]** (ジョブ)ビューで、縦に並んだタブの**[Extra]** (追加)を選択し、**[Implicit tContextLoad]** (暗黙的なtContextLoad)チェックボックスをオンにして、ジョブで明示的に**tContextLoad**コンポーネントを使うことなくコンテキストをロードするようにします。
2. コンテキストパラメーターのロード元を選択します。コンテキストソースは、2カラム構成のフラットファイルでも、2カラム構成のデータベーステーブルでも構いません。このユー

スケースでは、データベース接続情報をデータベーステーブルに格納しているので、**[From Database]** (データベースから)オプションを選択します。

3. データベース入力コンポーネントの基本設定を定義するように、データベース接続情報を定義します。

   この例では、すべての接続パラメーターをこの特定のジョブに対してのみ使用するので、**[Property Type]** (プロパティタイプ)リストから**[Built-In]** (組み込み)を選択して、接続情報を手動で入力します。



4. **[Table Name]** (テーブル名)フィールドに、**[Contexts]** (コンテキスト)ビューで定義したdb_connectionというコンテキスト変数を入力します。これで、ジョブの実行時にコンテキストパラメーターを動的にロードするデータベーステーブルを選択できるようになります。
5. データベーステーブルからすべての接続情報を無条件でフェッチするため、**[Query Condition]** (クエリ条件)フィールドは空白のままにします。
6. **[Print operations]** (操作の出力)チェックボックスをオンにして、ジョブの実行時にロードされるコンテキストパラメーターがリスト表示されるようにします。

ジョブを実行する

手順

1. **Ctrl+S**を押してジョブを保存し、**F6**を押してジョブを実行します。
2. ダイアログボックスが開いて、データベースを選択するように指示されます。データベースを選択したら、**[OK]**をクリックします。

ロードされたコンテキストパラメーターと選択したデータベースの"employees"テーブルのコンテンツが**[Run]** (実行)コンソールに表示されます。

3. ここで**F6**を押すと、ジョブが再起動するので、プロンプトメッセージに応じて、もう一つのデータベースを選択します。

   ロードされたコンテキストパラメーターともう一つのデータベースの"employees"テーブルのコンテンツが**[Run]** (実行)コンソールに表示されます。

マルチスレッド実行フィーチャーを使ったジョブの並列実行

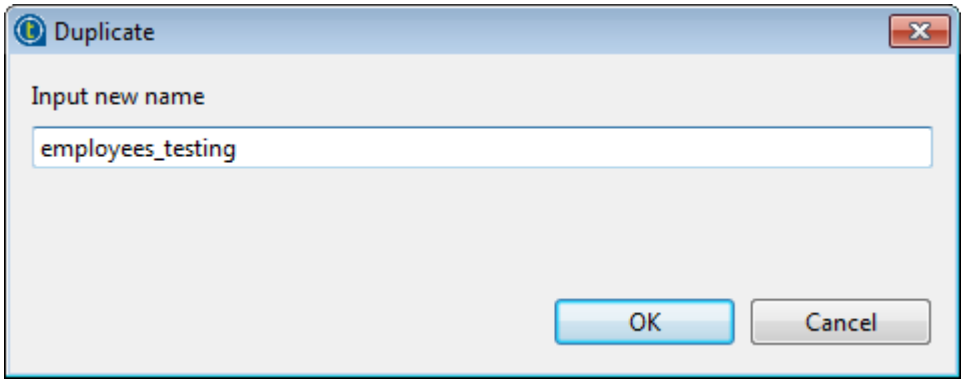
このユースケースでは、前述のユースケース出力ストリームフィーチャーの使用（14ページ）をベースにして、マルチスレッド実行フィーチャーを使って2つのジョブを並列実行し、テスト環境と本番環境の両方で同時に従業員情報を表示する方法を見ていきます。 処理データが大量である場合は、このフィーチャーにより、Talend Studioのジョブ実行パフォーマンスを大幅に改善することができます。

マルチスレッド実行フィーチャーの詳細は、「 **複数のサブジョブを並列に実行する方法** 」(https://help.talend.comのTalend Studioユーザーガイド)を参照して下さい。

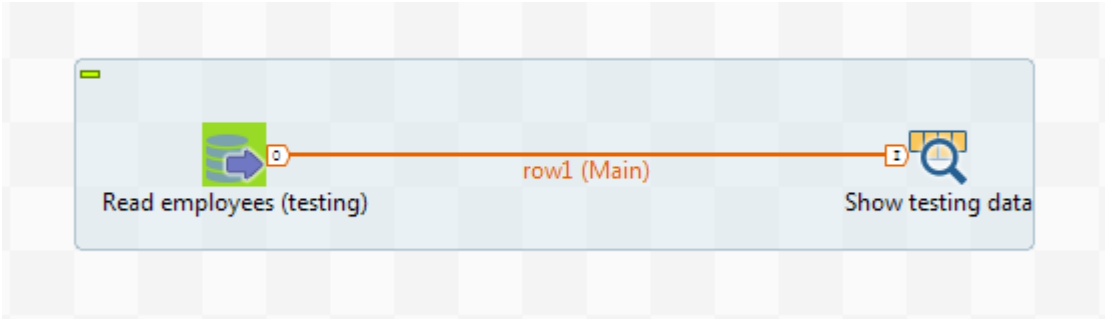
複数のコンテキストの従業員データを読み込むようにジョブを設定する

手順

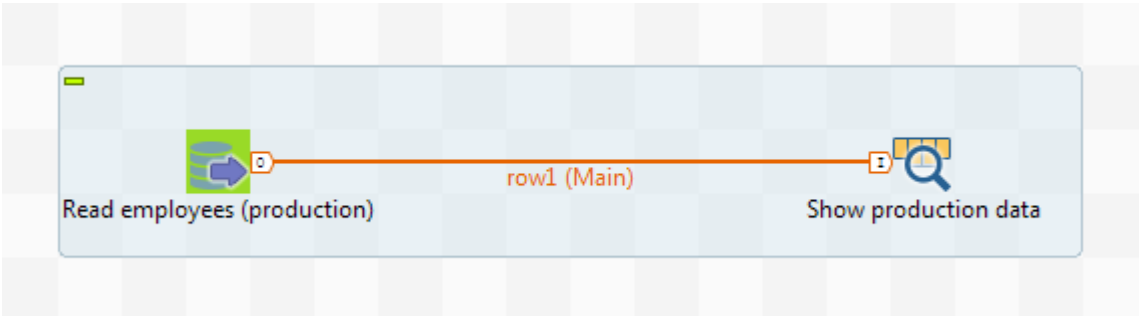
1. **[Repository]** (リポジトリ)ツリービューで、ユースケース出力ストリームフィーチャーの使用（14ページ）で作成したジョブを右クリックして、コンテキストメニューから**[Duplicate]** (複製)を選択します。次に、**[Duplicate]** (複製)ダイアログボックスに新しいジョブ名のemployees_testingを入力し、**[OK]**をクリックします。



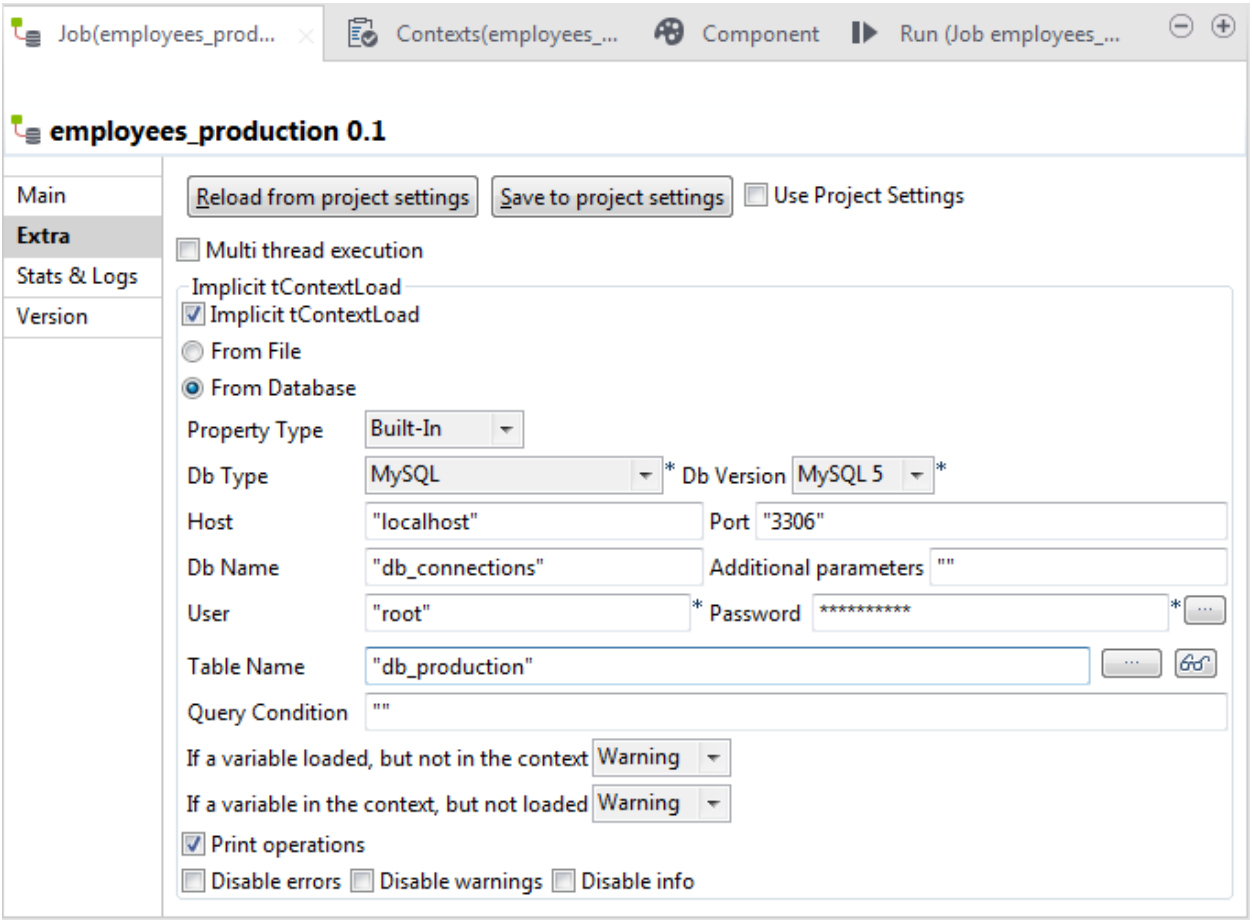
2. 新しいジョブを開いて、コンポーネントに、それぞれの役割に基づいたラベルを付けます。



3. この手順を繰り返して、employees_productionという名前のジョブをもう一つ作成します。



4. 両方のジョブの**[Contexts]** (コンテキスト)ビューで、**db_connection**変数を削除します。
5. ジョブ**employees_testing**の**[Job]** (ジョブ)ビューの**[Extra]** (追加)タブで、データベース設定の**[Table Name]** (テーブル名)フィールドに`db_testing`と入力し、ジョブ**employees_production**の**[Job]** (ジョブ)ビューの**[Extra]** (追加)タブで、**[Table Name]** (テーブル名)フィールドに`db_production`と入力します。

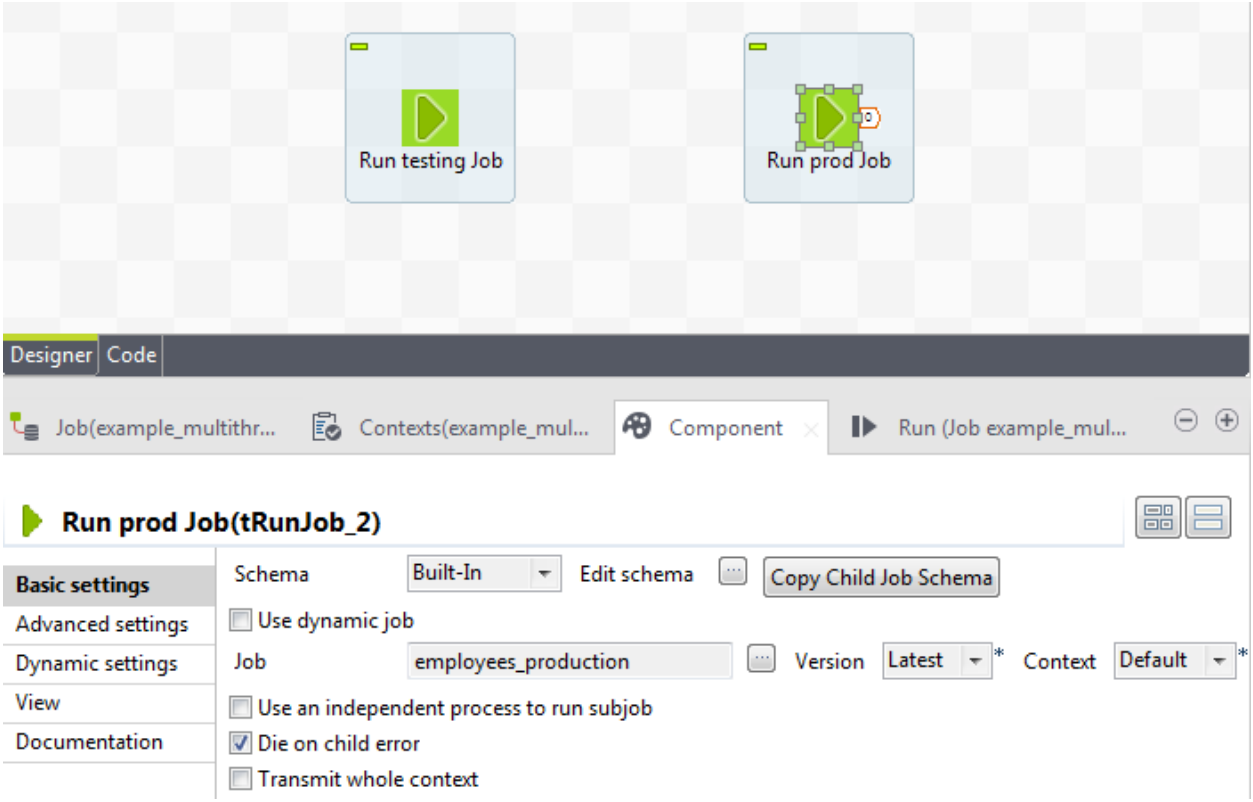


ジョブを並列実行するように親ジョブをセットアップする

手順

1. 新しいジョブを作成し、デザインワークスペース上で2つの**tRunJob**コンポーネントを追加して、各コンポーネントにその役割に基づいたラベルを付けます。

2. 最初の**tRunJob**コンポーネントの**[Component]** (コンポーネント)ビューで、**[Job]** (ジョブ)フィールドの横にある**[...]**ボタンをクリックして、実行するジョブとして`employees_testing`を指定します。
3. もう一つのジョブ`employees_production`を実行するように2つ目の**tRunJob**コンポーネントを設定します。



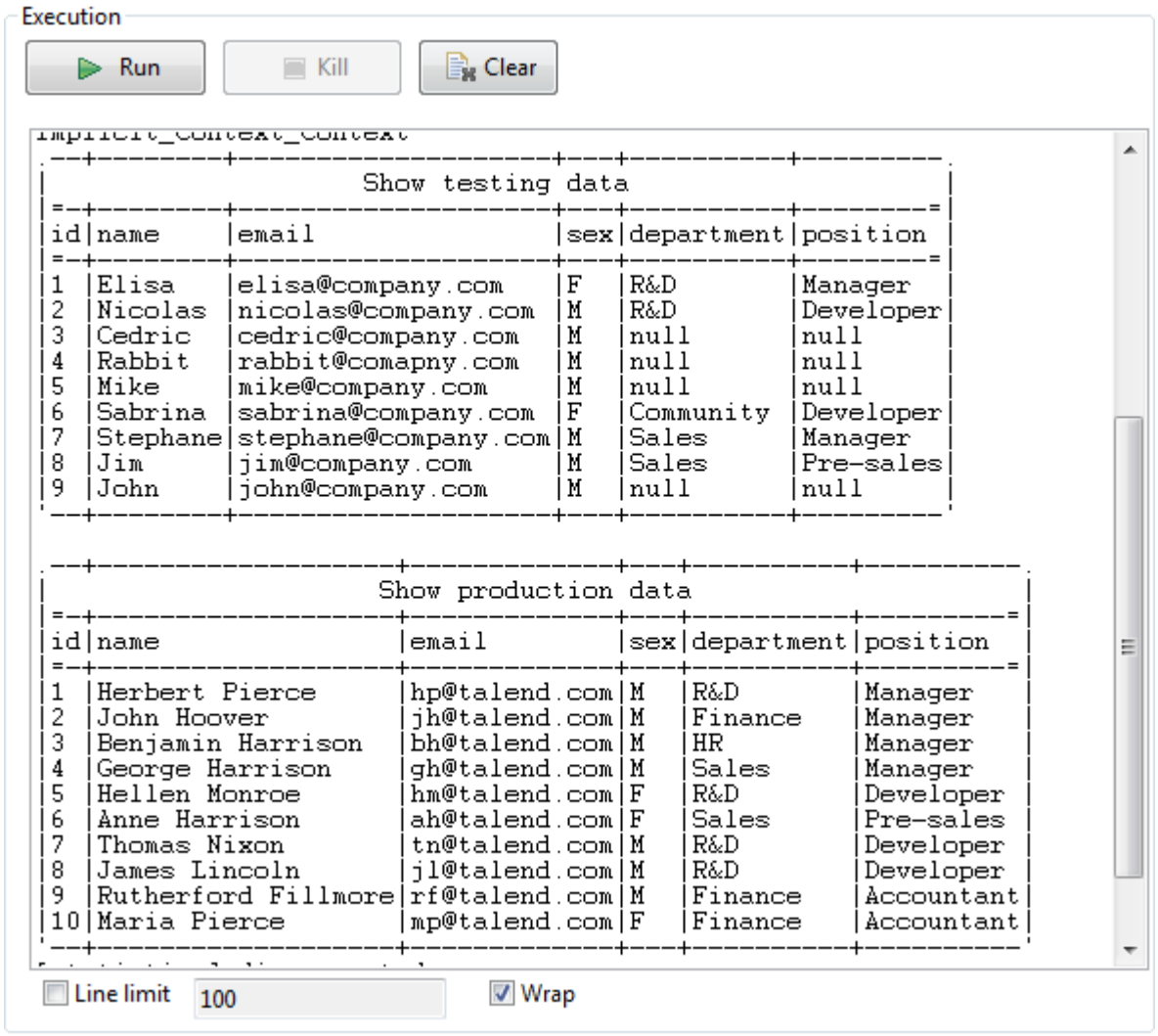
4. **[Job]** (ジョブ)ビューの**[Extra]** (追加)タブで、**[Multi thread execution]** (マルチスレッド実行)チェックボックスをオンにして、マルチスレッド実行フィーチャーを有効にします。

ジョブを実行する

手順

1. **Ctrl+S**を押して各ジョブを保存します。
2. 親ジョブで、**F6**を押すか、**[Run]** (実行)ビューの**[Run]** (実行)をクリックして、子ジョブの実行を開始します。

   子ジョブが並列実行され、両方のデータベースから従業員データが読み込まれ、コンソールにデータが表示されます。


Execution
Run
Kill
Clear
Show testing data
id|name|email|sex|department|position
1|Elisa|elisa@company.com|F|R&D|Manager
2|Nicolas|nicolas@company.com|M|R&D|Developer
3|Cedric|cedric@company.com|M|null|null
4|Rabbit|rabbit@comapny.com|M|null|null
5|Mike|mike@company.com|M|null|null
6|Sabrina|sabrina@company.com|F|Community|Developer
7|Stephane|stephane@company.com|M|Sales|Manager
8|Jim|jim@company.com|M|Sales|Pre-sales
9|John|john@company.com|M|null|null
Show production data
id|name|email|sex|department|position
1|Herbert Pierce|hp@talend.com|M|R&D|Manager
2|John Hoover|jh@talend.com|M|Finance|Manager
3|Benjamin Harrison|bh@talend.com|M|HR|Manager
4|George Harrison|gh@talend.com|M|Sales|Manager
5|Hellen Monroe|hm@talend.com|F|R&D|Developer
6|Anne Harrison|ah@talend.com|F|Sales|Pre-sales
7|Thomas Nixon|tn@talend.com|M|R&D|Developer
8|James Lincoln|jl@talend.com|M|R&D|Developer
9|Rutherford Fillmore|rf@talend.com|M|Finance|Accountant
10|Maria Pierce|mp@talend.com|F|Finance|Accountant
Line limit
100
Wrap