実習 6 隣接するポリゴンの融合・フィーチャーの集約

実習 5 と同じソースデータ（市区町村区域ポリゴン、平成 27 年国勢調査市区町村別人口等テーブル）を使用して、都道府県の区域のポリゴンを作成し、その属性に都道府県別の人口等を付加するためのワークスペースを作成します。実行結果は FME Data Inspector で確認します。

ソースデータ

mmm20151001.shp	Esri Shapefile 形式 市区町村区域ポリゴン
H27Kokuchou.csv	CSV 形式 平成 27 年国勢調査市区町村別人口等テーブル

今回のワークスペースは、次の 3 つのことを行わなくてはなりません。

1) 市区町村ポリゴンを都道府県ごとに融合・集約して、都道府県ポリゴンを作成する。
2) 市区町村別人口等テーブルに基づき、都道府県別に人口等を集計する。
3) 都道府県ポリゴンに、都道府県別の人口等の集計結果を結合する。

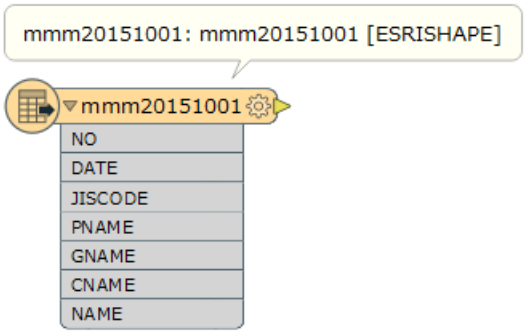
1)と 2)を部分的に独立したデータフローと考え、3)で FeatureMerger によってそれらのデータフローから送られてきたフィーチャーを結合する方針とします。

FeatureMerger は実習 5 でも使用しましたが、今回は、都道府県ポリゴンと都道府県別の人口等の集計結果を結合するための共通の属性が必要になります。
ソースデータの市区町村ポリゴン、人口等テーブルはどちらも属性として市区町村コードを持っているので、市区町村コードに基づいて都道府県コード（01:北海道～47:沖縄県）を作成し、3)ではそれが一致することを条件として都道府県ポリゴンに人口等の集計結果を結合することにします。

6-1. 都道府県ポリゴン作成のためのデータフロー

(1) リーダーの追加

これまでの実習と同じ手順により、Esri Shapefile 形式の市区町村区域ポリゴンを読み込むためのリーダーとリーダーフィーチャータイプをワークスペースに追加してください。



(2) 都道府県コードの作成と不要な属性の削除

前述したように、後で人口等の集計結果を結合するときに都道府県コードを使用する方針なので、あらかじめ、市区町村コードに基づいて都道府県コードを新たな属性として作成しておきます。

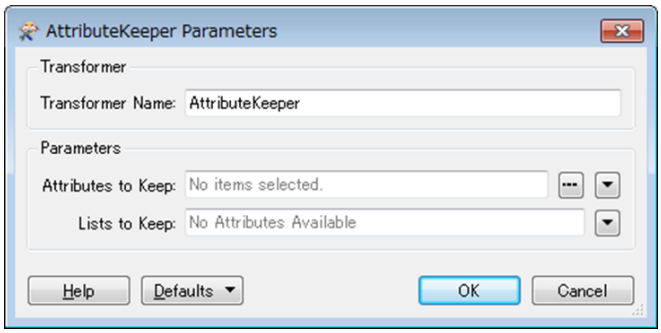
1) StringFormatter: JISCODE（市区町村コード）を 5 桁文字列に変更します（北海道～栃木県の市区町村コードの先頭に 0 を付加）。パラメーターの設定方法は実習 3 を参照してください。

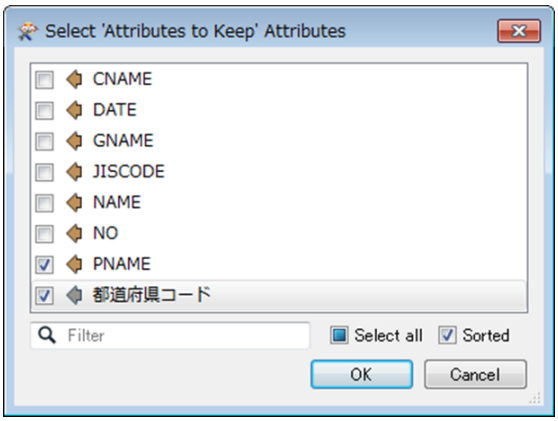
2) SubstringExtractor: 書式設定後の JISCODE（市区町村コード）の先頭 2 文字（都道府県コード）を抽出し、「都道府県コード」属性に格納します。パラメーターの設定方法は実習 5 を参照してください。

また、市区町村ポリゴンを都道府県ポリゴンに変換すると、都道府県名（PNAME）と都道府県コード以外の属性は意味がなくなるので、それらを削除しておきます。

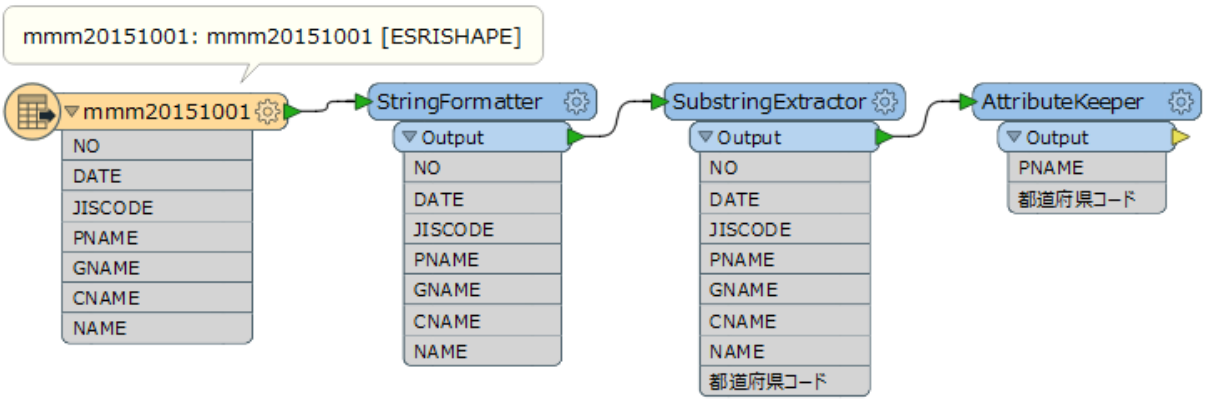
属性を削除するには、AttributeRemover（パラメーターで指定した属性を削除する）、AttributeKeeper（パラメーターで指定した属性を残す）トランスフォーマーなどが使用できます。
ここでは削除せずに残す属性の方が少ないので AttributeKeeper を使うことにします。

AttributeKeeper を SubstringExtractor に接続し、パラメーター設定画面で Attributes to Keep 右側の[…]ボタンをクリックするとチェックボックスつきリストで属性が表示されるので（右図）、残すべき属性として PNAME（都道府県名）と都道府県コードを選択してください。





ここまでのデータフローは、次のようになります。



データフローの途中で不要な属性を削除することは、必須ではありません。しかし、明らかに不要であることが分かっている属性が数多くある場合、それらを削除することによってキャンバス上の表示内容がすっきりしてワークスペースの作成や保守がしやすくなるとともに、実行時に FME がそれらを保持したりコピーしたりする必要がなくなるため、効率も良くなるという効果もあります。

(3) ポリゴンの融合 Dissolver

境界を接して隣り合う、または、重なり合う面の境界を削除してひとつの面に変換することを「融合する（dissolve）」と言い、それを行うには Dissolver トランスフォーマーを使用します。

現実世界では、隣接する市区町村間の境界は隙間や重なりがなくぴったりと接しているはずですが、コンピューターの処理ではごくわずかな計算誤差を避けることができないため、データ上の境界が厳密に一致しているとは限りません。使用しているソースデータの市区町村区域ポリゴンデータにも、データ作成過程で生じた計算誤差と見られる極めて小さなズレのある場所があります。

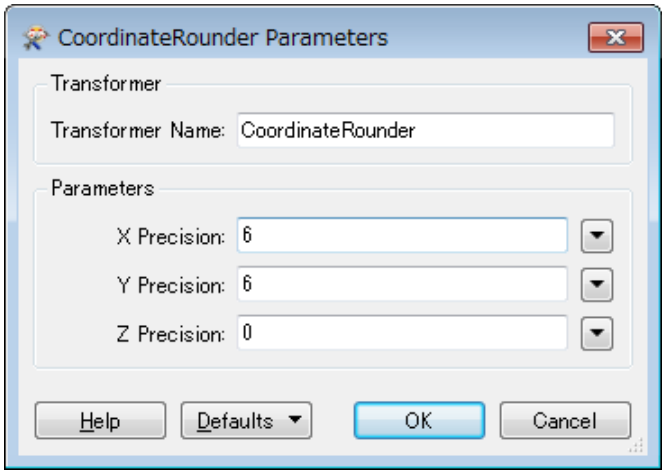
Dissolver で境界が接する面を融合するとき、そのような微小なズレがある場合には処理に非常に長い時間がかかることがあります。また、融合後の面では、元の隣接する面の間の微小な隙間が「穴」になり、望ましい結果が得られないこともあります。

これを回避する方法は実際のデータの状態によって異なりますが、ここでは、ポリゴンを構成する全ての頂点座標値の小数部を 6 桁に丸めることによってデータ上の誤差を吸収してから、Dissolver を適用することにします。

1) CoordinateRounder の接続

ジオメトリの頂点座標値は、CoordinateRounder によって指定した桁数で丸めることができます。CoordinateRounder を AttributeKeeper に接続し、パラメーターを次のように設定してください。

X Precision: 6（X 座標値の小数部桁数）
Y Precision: 6（Y 座標値の小数部桁数）
Z Precision: 0（市区町村ポリゴンデータは Z 座標を持たないので不使用）



注 1: 座標値の内部表現は浮動小数点数であるため、小数部の桁数が数学的に厳密に設定できるわけではありませんが、同じ桁数で全ての座標値を丸めればそれに伴う計算誤差も同じになるので、誤差によってずれていた座標の値を一致させることができます。

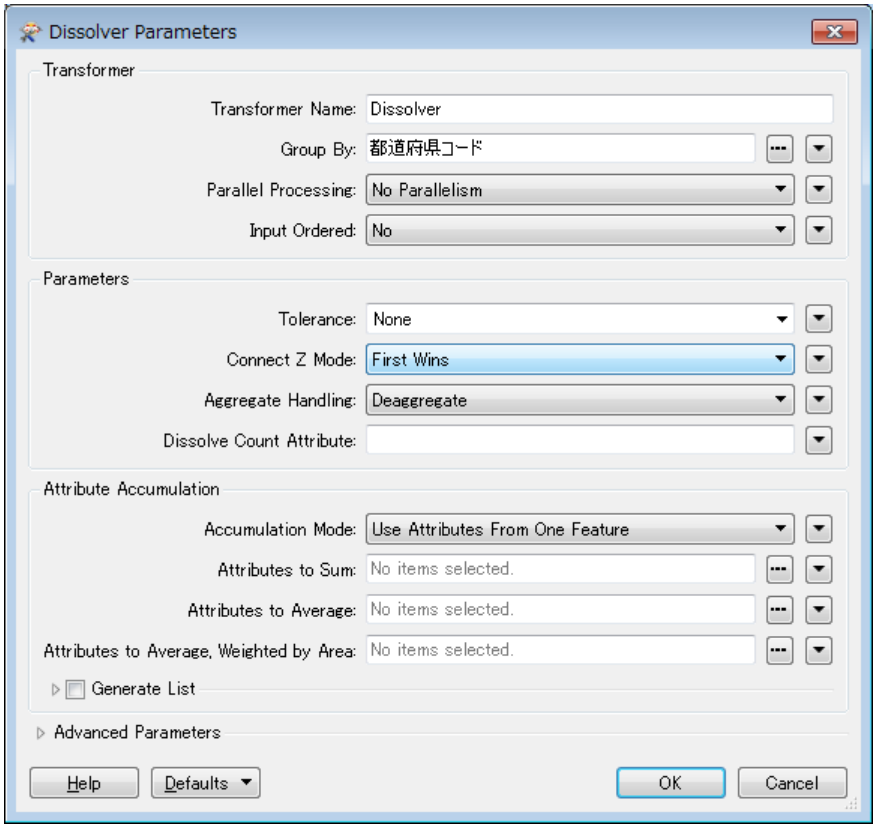
注 2: Dissolver を使用する前に常にこのような前処理が必要であるとは限りません。また、前処理が必要な場合でも、CoordinateRounder が妥当であるとも限りません。前処理の要否、要する場合の方法はソースデータの状態に依存しており、実務では FME Data Inspector でデータ内容を確認しながら試行錯誤によって決定する必要があります。

注 3: CoordinateRounder の X/Y/Z Precision パラメーターに 0 か負の値を設定すると、座標値を整数部で丸められます。例えば、0 なら整数、-1 なら 10 の位、-2 なら 100 の位で丸められます。

2) Dissolver の接続

Dissolver は、入力フィーチャー（面）に対して融合処理を行い、融合後のフィーチャーを Area ポートから出力します。
Dissolver を CoordinateRounder に接続し、パラメーターを次のように設定してください。

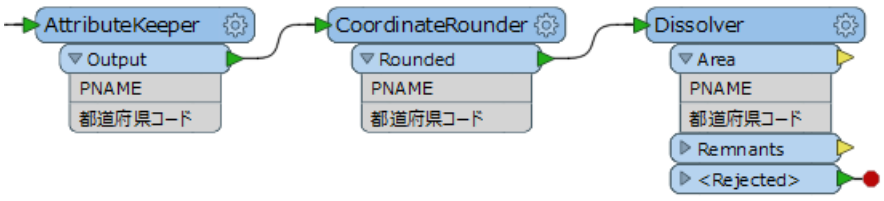
Group By: 都道府県コード
Accumulation Mode: Use Attributes From One Feature



重要なのは Group By パラメーターです。ここに何も設定しなければ、入力されたポリゴンのうち境界を接して隣り合う、または重なり合うものが全て融合されます。ここに"都道府県コード"を設定することにより、同じ都道府県コードを持つフィーチャーがグループ化され、グループ内、すなわち、同じ都道府県に属する市区町村の間だけで融合処理が行われます。

Group By パラメーターに設定された属性は、融合後のフィーチャーに引き継がれます。その他の属性（この場合は PNAME: 都道府県名）の取り扱いは、Accumulation Mode パラメーターの設定によって異なり、上のように "Use Attribute From On Feature" を選択した場合は、グループ内のフィーチャーのうち、どれかひとつが持っていた属性が引き継がれます。
この場合は、都道府県コードでグループ化したので、グループ内のどれかひとつの PNAME（都道府県名）を融合後のフィーチャーに引き継ぐことで、問題ありません。

その他のパラメーターの設定は、デフォルトのままで構いません。AttributeKeeper 以降のデータフローは、次のようになります。



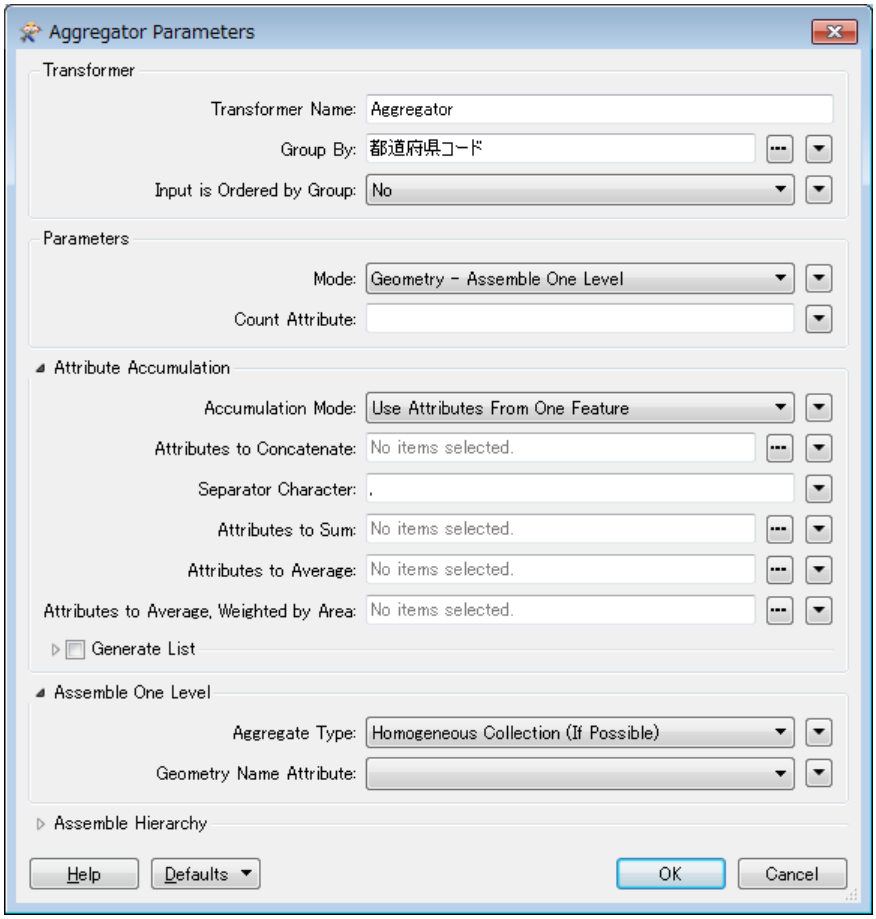
(4) フィーチャーの集約 Aggregator

Dissolver はシングルパートポリゴン（1 フィーチャー1 ポリゴン）に対して処理を行い、処理結果もシングルパートポリゴンになります。入力されたポリゴンがマルチパートポリゴンの場合は、シングルパートに分解してから処理が行われます。
ソースデータの市区町村ポリゴンは、離島や飛び地も市区町村ごとにまとめたマルチパートポリゴンですが、Dissolver による処理後は、都道府県に離島や飛び地の領域があれば、それらはすべて分離されたシングルパートポリゴンとなります。
都道府県の単位で人口等の集計結果を結合するときにそれでは不都合ですから、都道府県ごとにまとめたマルチパートポリゴンに変換しておきます。

複数のフィーチャーをひとつのフィーチャーにまとめることを「集約する（aggregate）」と言い、それを行うには Aggregator トランスフォーマーを使用します。
この場合は、都道府県コードが同じフィーチャーを集約することによって、都道府県ごとのマルチパートポリゴンに変換することができます。

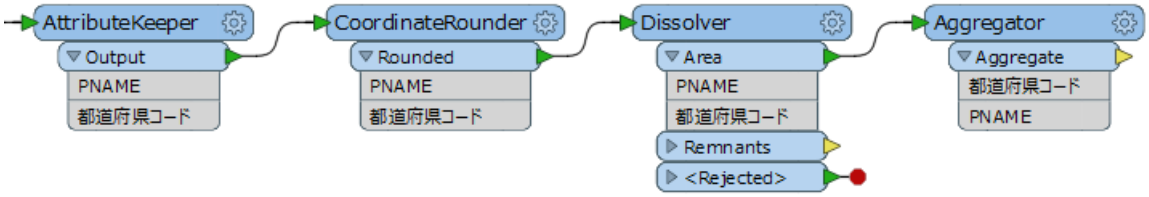
Aggregator を Dissolver の Area ポートに接続し、次のようにパラメーターを設定してください。

Group By: 都道府県コード
Mode: Geometry – Assemble One Level
Accumulation Mode: Use Attributes From One Feature



ここでも、Group By パラメーターの設定が重要です。ここに"都道府県コード"を指定することにより、入力フィーチャーが都道府県ごとに集約されることになります。
Mode パラメーターでは集約の仕方を指定します。"Geometry – Assemble One Level"を指定することで、一般的なマルチパートジオメトリに変換されます。
Accumulation Mode パラメーターについては、Dissolver と同じです。

その他のパラメーターの設定はデフォルトのままで構いません。



ここまでで、都道府県ポリゴン（マルチパート）を作成するためのデータフローができました。Inspector を Aggregator に接続、実行して結果を確認しておきましょう。

ポリゴンの融合はやや時間のかかる処理です。コンピューターの性能により異なりますが、数十秒程度かかることもあります。

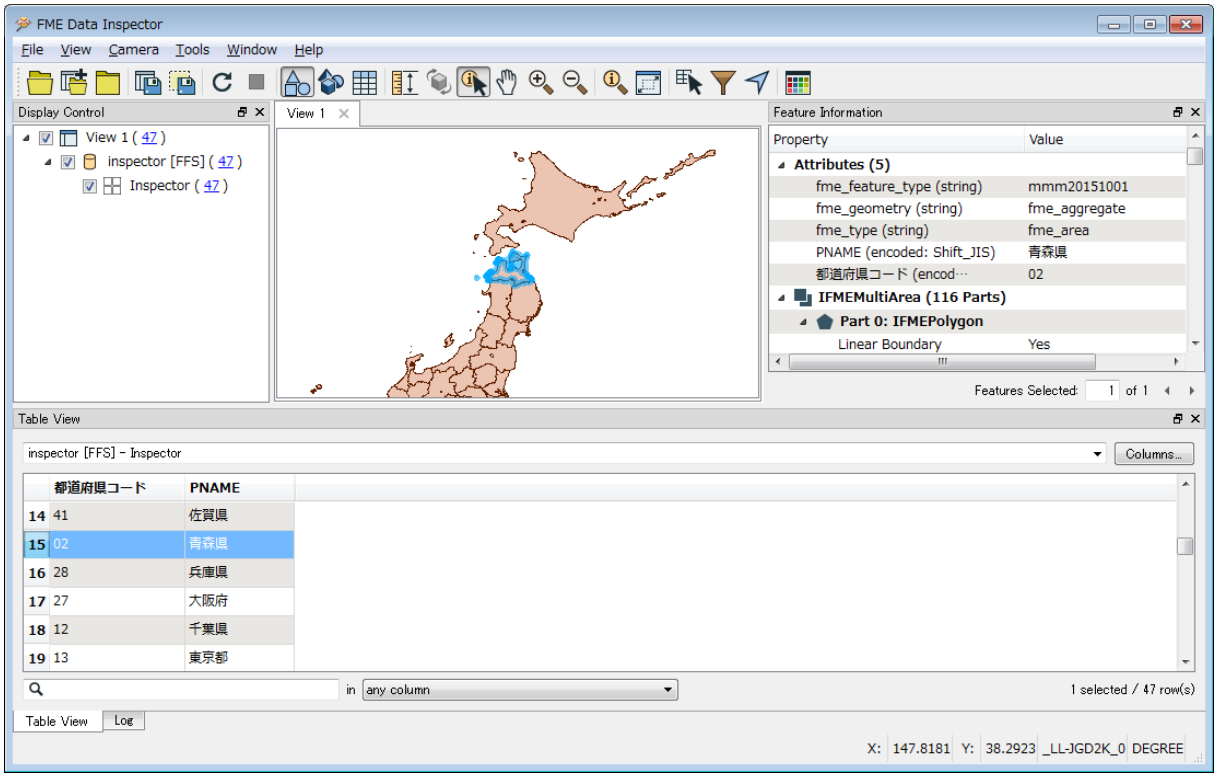


Table View で、行の並び順（フィーチャーの出力順に等しい）が都道府県コード順ではないことに注意してください。

実習 5 までのように、読み込んだフィーチャーをひとつずつ処理するワークスペースでは、ソースデータからのフィーチャーの読込順（ソースデータに記述されている順番）がデータフローの途中で変わることはありません。

しかし、Dissolver や Aggregator のように複数のフィーチャーをまとめて処理するトランスフォーマーを経由した場合、その後の順番は一般に予測できません。

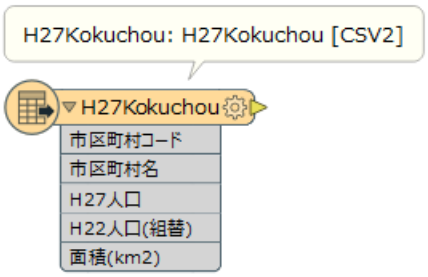
ただし、Dissolver、Aggregator ともに、Group By パラメーターで指定した属性の値が等しいフィーチャー（同一グループに属するフィーチャー）は連続して入力されることが確かである場合は、Input Ordered パラメーターを"By Group"に設定（または、Input Is Ordered By Group パラメーターを"Yes"に設定）することができます。その結果、変換後のフィーチャーの出力順がグループごとの入力順と同じになり、また、実行時の効率が良くなる可能性もあります。

6-2. 都道府県別人口等集計のためのデータフロー

都道府県ポリゴンを作成するためのデータフローと同じワークスペースで、都道府県別の人口等を集計するためのデータフローを作成します。

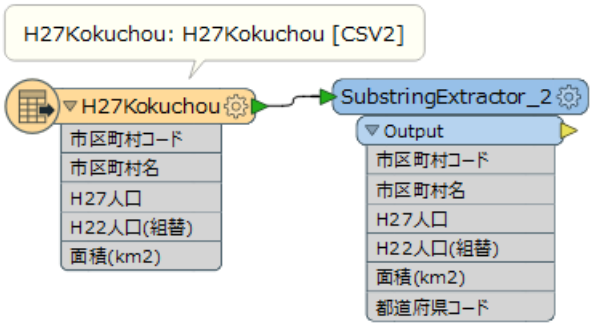
(1) リーダーの追加

実習 5 と同じ手順により、CSV 形式の国勢調査人口等テーブルを読み込むリーダーとリーダーフィーチャータイプをワークスペースに追加してください。キャンバス上のリーダーフィーチャータイプオブジェクトの位置は、都道府県ポリゴン作成のためのデータフローの下にします。



(2) 都道府県コードの作成と不要な属性の削除

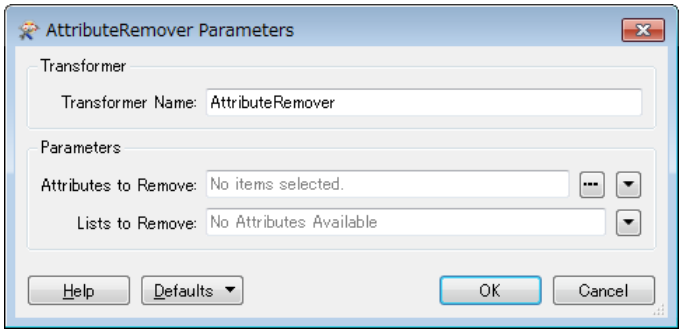
StringConcatenator を追加、接続し、市区町村コードに基づいて都道府県コードを作成します。
国勢調査人口等テーブルの市区町村コードは 5 桁の文字列（北海道～栃木県のコード先頭には 0 が付加されている）なので、書式の設定をしなくても SubstringExtractor によって市区町村コードの先頭 2 文字を抽出し、新たな属性として都道府県コードを作成することができます。

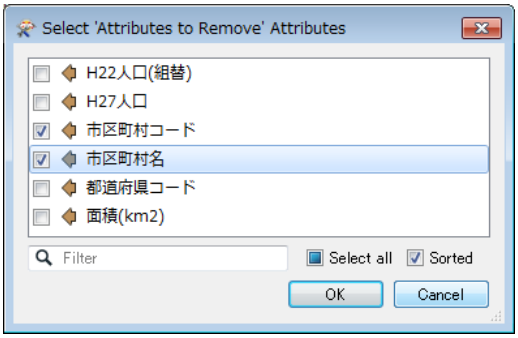


ひとつのワークスペース内（厳密にいえば、ひとつのキャンバス上）では、トランスフォーマーオブジェクトの名前（Transformer Name パラメーターの値）はユニークでなければなりません。トランスフォーマーを追加したとき、デフォルトではトランスフォーマー名がオブジェクトの名前となりますが、同じ名前のオブジェクトがすでにワークスペース内にあるときは、自動的にアンダースコアをはさんだ枝番がつきます。
この場合、都道府県ポリゴン作成用のデータフローですでに SubstringExtractor を使用しているので、ここで追加した SubstringExtractor のオブジェクト名は SubstringExtractor_2 となりました。
オブジェクト名（Transformer Name パラメーターの値）は、必要に応じて、パラメーター設定画面で編集できます。

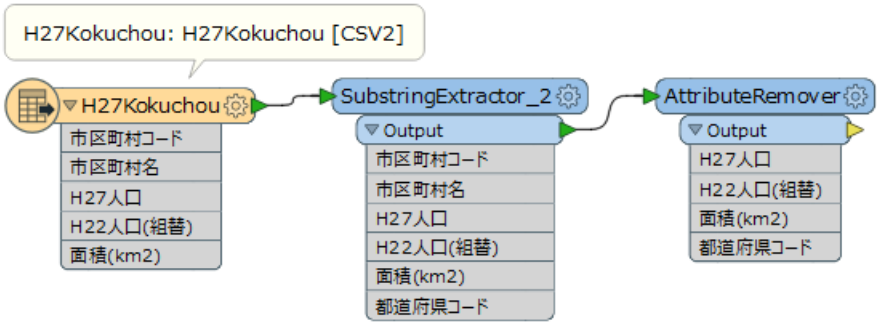
次に、これ以降の処理では市区町村コードと市区町村名は不要ですから、削除しておきます。削除する属性は 2 つだけなので、ここでは AttributeRemover を使うことにします。

AttributeRemover を SubstringExtractor に接続してからパラメーター設定画面を開き、Attribute to Remover パラメーターで削除する属性を指定してください。





ここまでのデータフローは次のとおりです。



(3) 属性の集計 Aggregator

Aggregator は、複数のジオメトリを集約してマルチパートにするだけでなく、集約したフィーチャーの属性値の集計などを行う機能も持っています。
その機能を使い、市区町村別の H27 人口、H22 人口(組替)、面積(km2)を都道府県別に集計します。

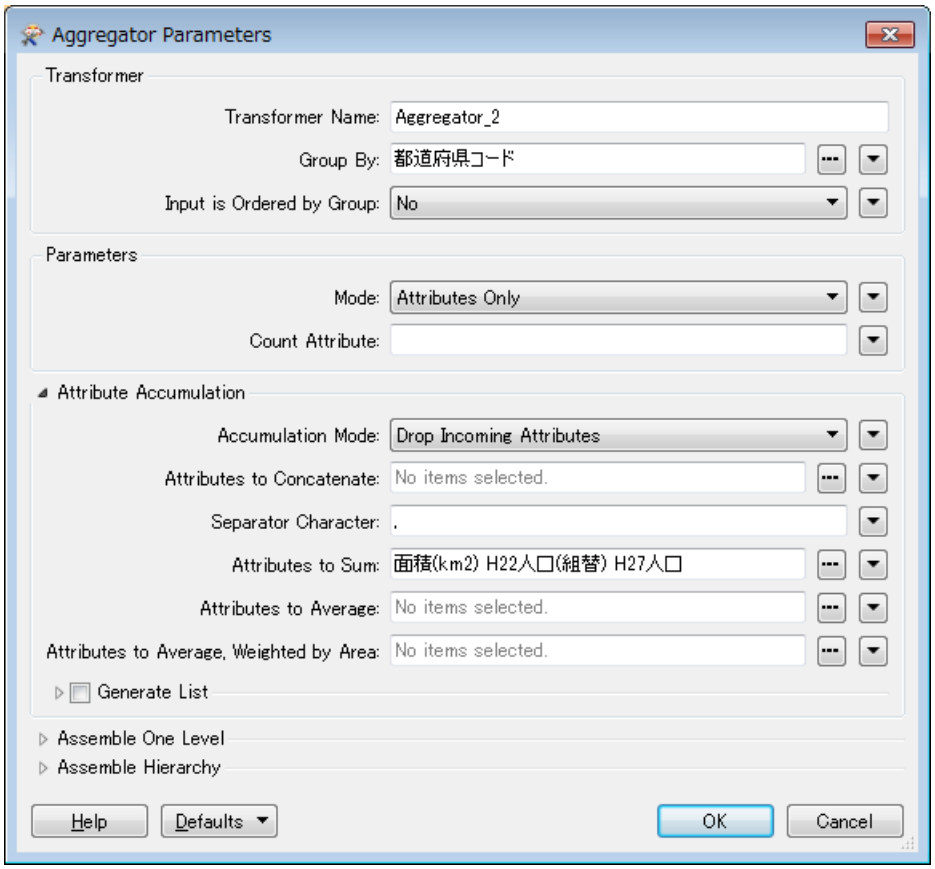
Aggregator を追加して AttributeRemover に接続し、パラメーターを次のように設定してください。

Group By: 都道府県コード
都道府県ごとに集計するので、Group By に都道府県コードを設定します。

Mode: Attributes Only
国勢調査人口等テーブルから読み込まれたフィーチャーにはジオメトリはないので、Mode は Attributes Only（属性のみ）とします。

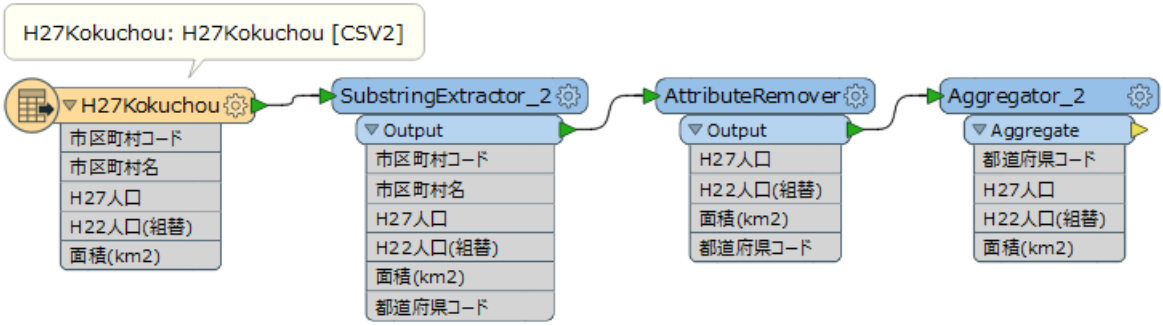
Attributes to Sum: H27 人口, H22 人口(組替), 面積(km2)
Attributes to Sum パラメーターに、グループごとに集計して合計値を求める属性を指定します。パラメーター入力フィールド右側の[...]ボタンをクリックするとチェックボックスつきのリストによって属性が表示されるので、集計したい属性を選択してください。
集計結果は、ここで選択した属性名と同じ名前の属性に格納されます。

これらを設定した Aggregator のパラメーター設定画面は、次の図のようになります。



Aggregator 追加後のデータフローは次のようになります。

H27 人口等は、AttributeRemover まではソースデータから読み込まれた市区町村ごとの値であったものが、Aggregator で集約した後は、上記のパラメーター設定によってグループ = 都道府県ごとの合計値になることに留意してください。



Group By パラメーターは、Dissolver, Aggregator だけでなく、複数のフィーチャーを対象として処理を行うさまざまなトランスフォーマーにあります。FME を活用するうえで、「特定の属性値に基づいてフィーチャーをグループ化し、グループ単位で処理を行う」という Group By パラメーターの働きを理解することは不可欠です。

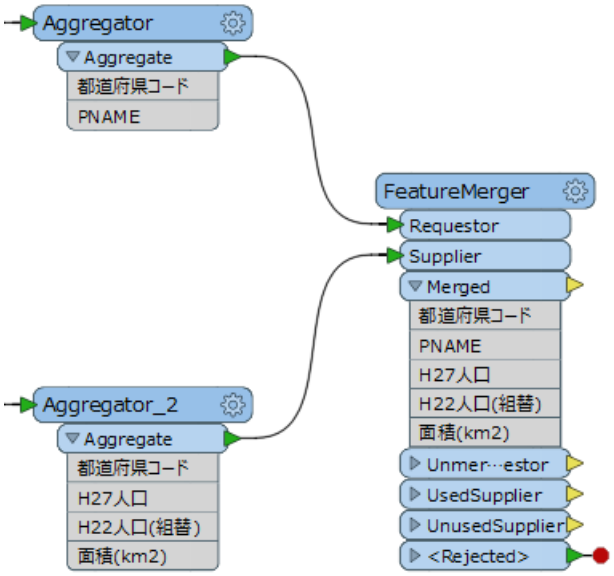
この実習で Group By パラメーターを設定したときの Dissolver, Aggregator の動作を確かめることにより、その理解を深めてください。

合計、平均値などを求めるには Aggregator を使用するのが簡便ですが、StatisticsCalculator トランスフォーマーを使用すると、それらに加えて、最大値、最小値、中央値、最頻値、標準偏差などの基礎的な統計値を求めることもできます。用途に応じて使い分けてください。

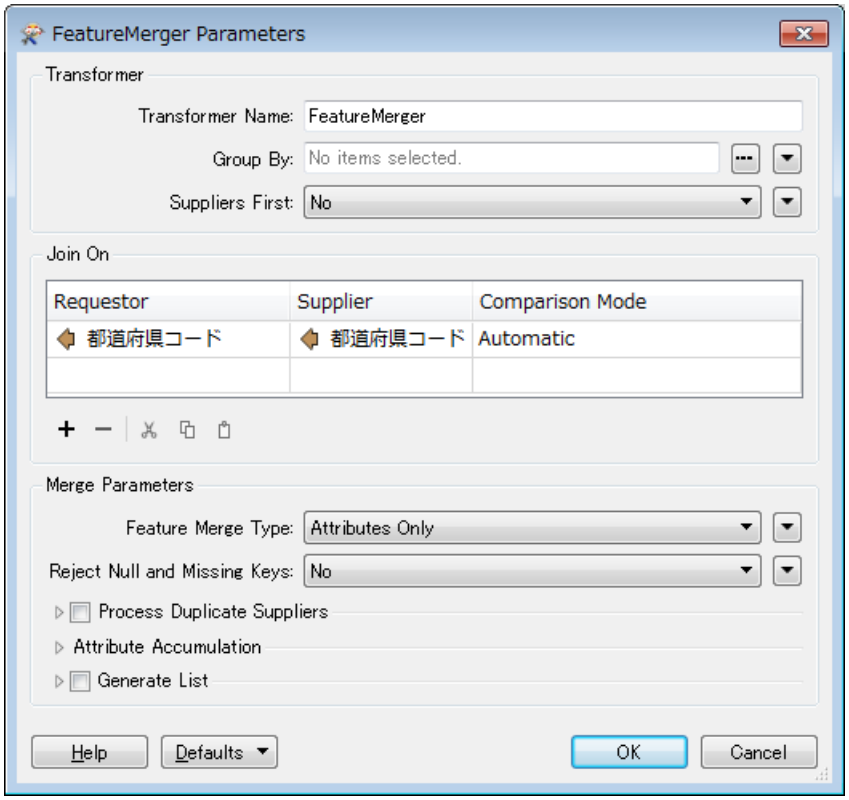
6-3. 属性の結合

2 つのデータフローを統合して、都道府県別の人口等の集計結果を対応する都道府県ポリゴンに結合します。それぞれのデータフローで「都道府県コード」属性を作成しておいたので、それが一致することを条件として FeatureMerger によって行います。

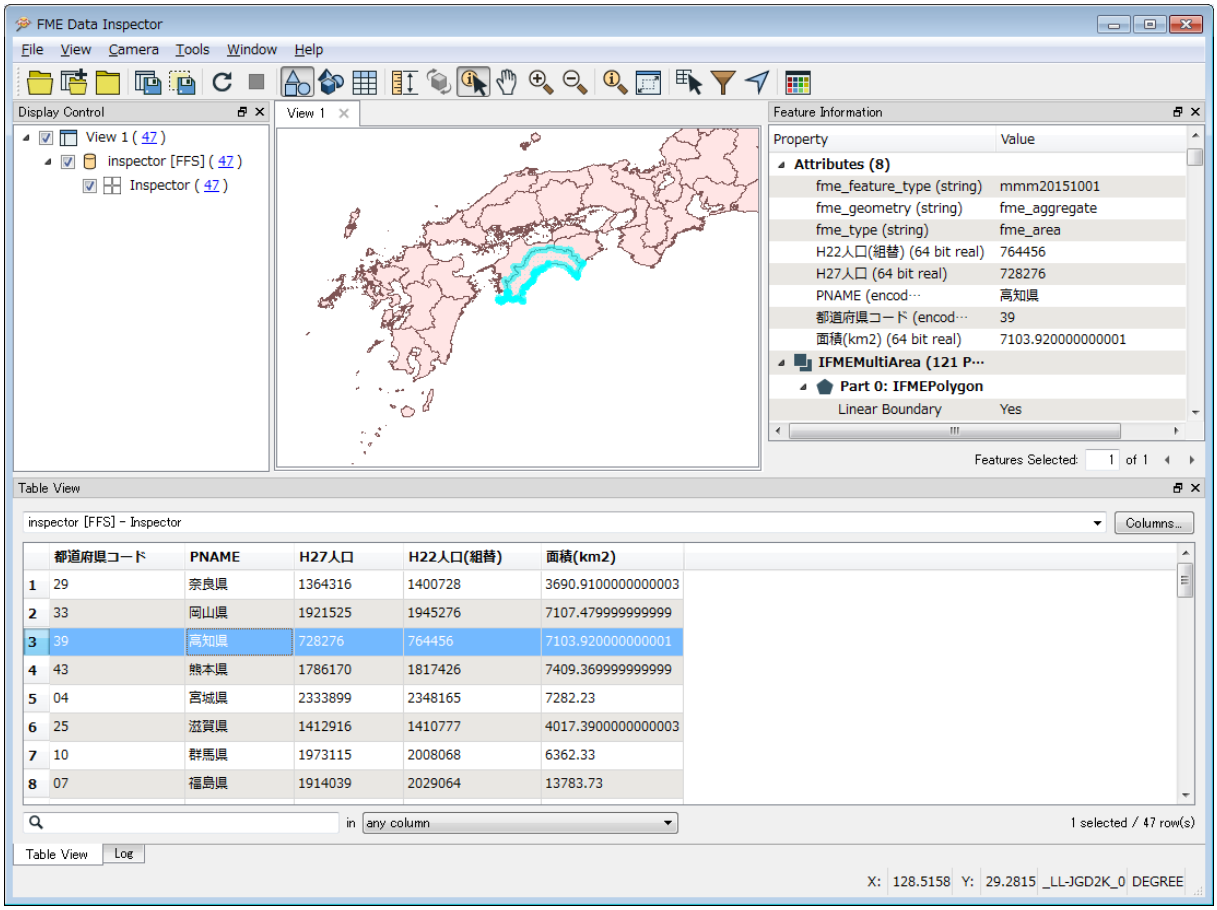
FeatureMerger をワークスペースに追加し、Requestor ポートに都道府県ポリゴンを作成するデータフロー最後の Aggregator、Supplier ポートに都道府県別の人口等を集計したデータフロー最後の Aggregator_2 を接続してください。



FeatureMerger のパラメーター設定は次のとおりです。



Inspector を FeatureMerger の Merged ポートに接続して実行し、変換結果を確認しましょう。



実習 6 はここまでです。主に次の事項を学びました。

・隣接するポリゴンの融合 Dissolver
・フィーチャーの集約（マルチパート化、属性の集計） Aggregator
・Group By パラメーターの働き
・属性の削除 AttributeRemover, AttributeKeeper

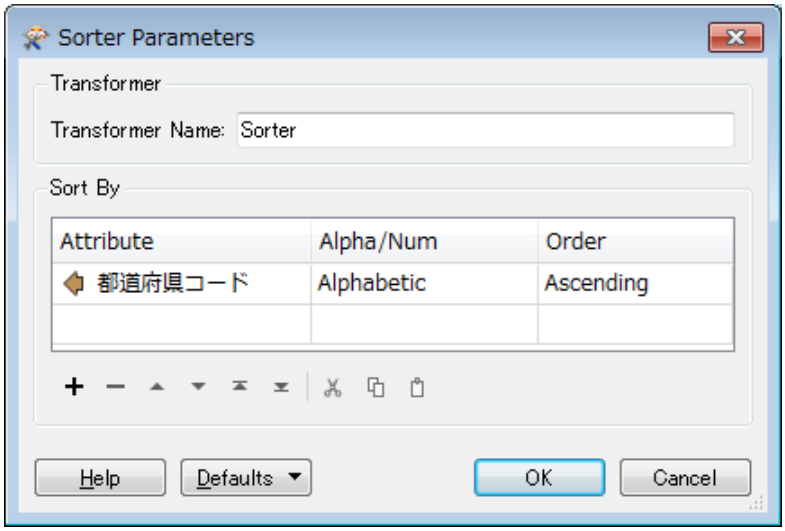
Sorter トランスフォーマー

出力先のデータセットが GIS 用のデータセットやデータベースである場合は、フィーチャーの格納順はあまり重要ではないかも知れません。

しかし、変換結果（属性）をテーブル形式で出力する場合など、一定の規則によってソートされていた方が良いこともあります。また、フィーチャーの入力順が特定の属性値によってソートされていることを前提とするトランスフォーマーもあります。

そのような場合は、Sorter トランスフォーマーによって特定の属性値の大小関係に基づいてフィーチャーの順番を並べ替えることができます。

例えば、この実習で作成したワークスペースの Inspector の前に Sorter を挿入し、そのパラメーターを次のように設定すると、フィーチャーの順番は都道府県コードの昇順で並べ替えられます。



Attribute: ソートキーとする属性
Alpha/Num: Alphabetic（辞書順）/ Numeric（数値順）
Order: Ascending（昇順）/ Descending（降順）

注 1: ソートキーは複数指定することができ、行の順番（上から下）が優先キーの順番となります。

注 2: Alphabetic は、厳密に言えば「文字コード（コンピューターが文字を識別するための数値）順」であり、属性値に日本語文字が含まれる場合は字義通りの「辞書順」になることは期待できません。