大量のソースデータを利用した浸水想定データ加工手法

嘉山陽一*

Processing method of　Inundation assumption data　using a large amount of source data

Yoichi Kayama*

I created a set of programs to create data for hazard maps that collect a large number of inundation assumption data, and to create aggregate data in quadratic mesh units to aggregate these data.
The data for the expected inundation areas are prepared by the national government, prefectures, and municipalities. I overlaid the 5-meter mesh with the data for each assumed inundation area to determine whether or not each mesh had the potential to inundate an area.
Each 5-meter mesh had an attribute value for the assumed inundation depth. However, a single 5-meter mesh may be included in the range of multiple assumed inundation data. In such cases, the mesh attribute is assigned the value of the greatest depth value among the assumed inundation depths of the assumed inundation areas where the 5-meter mesh crosses.　In addition, when overlay processing of inundation assumption data for rivers with large river basins was performed, the processing time became very long, and there were some data for which the work could not be completed in time. This paper describes the method used to speed up the long overlay process.

Keywords: Hazard map, Polygon overlay, Mass data processing,QGIS

１．はじめに

日本は地震や洪水等自然災害が多発する地域であるといえよう.河川の水害対策については多くの地点で浸水想定の計算が行われ市町村がハザードマップを作製して住民に配布するためのデータとして提供されている[1].しかしハザードマップで浸水想定区域に含まれていることが示されている地域でもその地域の住民がその情報を意識していなかったため実際に水害が起きた時に避難が遅れ大きな被害が発生するような事態も発生した[2].また近年不動産の取引を行う際に水害ハザードマップを利用して対象物件の水害リスクの説明が義務化された[3].

このように災害への対策で浸水想定データのような予測データの利活用が重要になってきている.また従来は紙で出力したものをユーザが見て判断することが主な用途であったが,様々な空間系オープンデータとそれの加工のためのソフトウェアが安易に入手できるようになってきたためデジタルの浸水想定データを他のデータと組み合わせてコンピュータ内で加工し,データから新たな情報を導き出すことが重要になってきている.ただし浸水想定区域データはデータ量が多いものもあり,方法によっては加工に多大な時間がかかることもある.本稿では NHK の全国ハザードマップ[4]で利用する浸水想定データを作成する作業で直面したデータ加工方法の問題点とそれへの対応方法について説明を行う.

２．データ加工作業概要

2.1．収集データ

国土地理院では「重ねるハザードマップ」という WEB サイトで浸水や土砂災害情報を電子地図として参照できる仕組みを提供している.しかしこの地図で利用している浸水想定データは国や自治体が作成したデータをお願いベースで収集しているので水防法で浸水想定を行うことが必要とされている「洪水予報河川」「水位周知河川」約 2200 本のうち 1770 本（2022 年 5 月時点）のデータしか地図化されていない[5].

*　正会員　朝日航洋株式会社（Aeroasahi corporation）
〒**350-1165**　埼玉県川越市南台南台 3-14-4　E-mail：**youichi-kayama@aeroasahi.co.jp**

NHK は国の各種機関や地方自治体から直接データを取得して網羅率の高いハザードマップの作製を試行した.そこで集められたデータは 34 テラバイト、1358 万 3715 本の Shape ファイルになった[5].これらのファイルを利用してハザードマップを作製する場合取捨選択や複数の浸水想定区域データが重なり合う区域でのデータ抽出加工が必要になる.それらの処理はデータ量が多い場合とてつもなく時間がかかるので,どのように空間演算を高速化するかの工夫が重要になる.

2.2. 作成したいデータ

今回は集めたデータを元に 5m メッシュデータとして浸水エリアに含まれているかどうかの判定されたデータを作成する.また浸水判定されたメッシュには想定浸水深のランク数値が格納されているものとする.この時ある 5m メッシュが複数の想定浸水深領域に含まれる場合はそれらの浸水深のうち一番深い数値が値として残されるものとする.

また 4 次メッシュ単位に浸水が想定される領域と想定されない領域の比率が数値として格納されたものを作成する.このデータには国勢調査等の統計データを結合して利用することを想定する（浸水想定区域内の想定人口等を算出可能にする）

2.3. メッシュ番号体系について

今回の作業の結果は 5m メッシュと 4 次メッシュというメッシュデータである.両方のメッシュとも標準メッシュから派生したメッシュではあるが派生方法に分岐があり別系統の枝分かれ結果であるため 5m メッシュのメッシュ番号に 4 次メッシュのメッシュ番号は含まれない.同系統のメッシュ番号の場合上位メッシュのメッシュ番号は下位メッシュのメッシュ番号の一部に含まれる.たとえば 1 次メッシュ 4830 の中の 2 次メッシュは 483013､483014 等になるので 2 次メッシュのメッシュ番号の先頭 4 桁はそのメッシュが含まれる 1 次メッシュのメッシュ番号になる.また 2 次メッシュ 483013 に含まれる 3 次メッシュは 48301348, 48301335 というようなメッシュ番号になる.この場合メッシュ番号の先頭 4 桁がその 3 次メッシュが所属する 1 次メッシュの番号であり,先頭 6 桁が所属する 2 次メッシュの番号になる.このように上位,下位の階層構造があるメッシュの体系内ではメッシュの番号によって上位,下位の包含関係を判定することができる.

このような上位下位体系のメッシュデータの場合は下位メッシュ単位で作成された統計データを上位メッシュ単位で再集計を行う場合は空間演算を使わないでメッシュ番号の部分文字列を利用した集計で行うことができる.そのような集計はデータベース管理システムや表計算ソフトウェアで計算することができる.

しかし 4 次メッシュと 5m メッシュは標準地域メッシュ(3 次メッシュ)を分割してできたものではあるが,5m メッシュの分割元は 4 次メッシュとは別系統の分割からの派生物であるためメッシュ番号だけによる包含関係判定はできない.

日本の標準地域メッシュは 1 次メッシュから 3 次メッシュまでを定義している基準地域メッシュと,3 次メッシュを分割して定義している分割地域メッシュが存在する[5].

表 1. 基準地域メッシュの区分方法

メッシュ名称	code	備考
第 1 次地域区画	4 桁	1 次メッシュ (緯度間隔 40 分,経度間隔 1 度)
第 2 次地域区画	6 桁	2 次メッシュ （緯度間隔 5 分 経度間隔 7 分 30 秒)
基準地域区画 （第 3 次地域区画)	8 桁	3 次メッシュ (緯度間隔 30 秒 経度間隔 45 秒)

ここの定義にある分割地域メッシュの 2 分の 1 地域メッシュが本稿で 4 次メッシュとしているメッシュであり,経度方向の長さが約 500ｍになる.国勢調査等のデータもこのメッシュ単位で公開されているものがあるので他の統計データと組み合わせて利用する場合に利用しやすいメッシュである.

表 2. 分割地域メッシュの区分方法

メッシュ名称	code	備考
1/2 地域メッシュ	9 桁	4 次メッシュ (緯度間隔 15 秒,経度間隔 22.5 秒)
1/4 地域メッシュ	10 桁	5 次メッシュ （緯度間隔 7.5 秒 経度間隔 11.25 秒)
1/8 地域メッシュ	11 桁	6 次メッシュ (緯度間隔 3.75 秒 経度間隔 5.625 秒)

5m メッシュも 3 次メッシュ（基準地域メッシュ）を元に複数回の分割を行ってできたメッシュであるが分割方法が分割地域メッシュとは異なる方法である.

表 3. 5m メッシュの分割定義

メッシュ名称	**code**	**備考**
3 次メッシュ(1Km メッシュ)	8 桁	基準地域メッシュ
100m メッシュ	10 桁	1km メッシュをタテヨコ 10 分割
50m メッシュ	12 桁	100m メッシュをタテヨコ 2 分割
10m メッシュ	12 桁	100m メッシュをタテヨコ 10 分割
5m メッシュ	14 桁	10m メッシュをタテヨコ 2 分割

よって 5m メッシュと 4 次メッシュ（2 分の 1 地域メッシュ）との包含関係は単純なメッシュ番号の部分文字列を用いて判定を行うことはできない.

5m メッシュは国土地理院が公開している基盤地図情報（数値標高モデル）[6]等の公開で利用されている.地形の細かい表現がこのメッシュで行われていることもあるので浸水想定区域の判定もこのメッシュが利用されている箇所が多い.

2.4. 当初のデータ加工方針

用意された浸水想定区域データは 5m 等のメッシュで用意されているものや,同一浸水深の連続領域で作成されたポリゴンとして用意されているようなものもある.また同じ領域に重なる複数の浸水想定区域のデータが存在する.これらの諸データを利用して 5m メッシュ単位の浸水想定区域への包含判定を行うと同時に複数の浸水想定がオーバーレイする場合それらのうちの最大浸水深の値を属性値として残す必要がある.そのためには各浸水想定区域と 5m メッシュのオーバーレイ判定を行い,その結果重なるデータの 5m メッシュ ID と想定浸水深の値が格納されたレコードを作業用のテーブルに出力する.その後そのテーブルからユニークな 5m メッシュ ID と最大の想定浸水深の値をペアにして取得することで最大想定浸水深つきの 5m メッシュデータが作成できると思われる.

また 4 次メッシュ単位の浸水区域割合については各 4 次メッシュ単位に浸水区域に重なる 5m メッシュの数を算出することで割合を算出することができる.単一の 4 次メッシュには 10000 の 5m メッシュが格納されるので 5m メッシュで想定浸水領域に重なるメッシュの数を 4 次メッシュ単位に集計すればよい.ただし前節で説明したとおり 4 次メッシュと 5m メッシュではメッシュ番号のみでメッシュの包含関係を簡単に示すことはできない.そのため作業用 5m メッシュデータを作成するときに 5m メッシュの各レコードにその 5m メッシュが含まれる 4 次メッシュの番号を属性値として保有させる.このテーブルを利用することによって 5m メッシュ単位の浸水区域重なり判定を行ったデータがあれば 4 次メッシュ単位の集計を行うことができる.

2.5. 利用ソフトウェアとデータ形式

以上のような加工処理を行うためのツールとデータ形式を選定しなければならない.大量のファイルに対する処理を行わなければいけないので多数のファイル名称を入力とするようなバッチ処理での利用が可能なツールであることが好ましい.

現在 Windows 版 QGIS は OSGeo4w shell というコマンド実行環境のもとでインストールされる[7].この環境では QGIS,GDAL,GRASS 等様々なオープンソース GIS パッケージを利用することができる.また Python 言語も利用できるようになっているので

Python でスクリプトを作成して実行することもできる.

QGIS は現在最も多く利用されているデスクトップ GIS である.各種 API を公開しているので Python から QGIS の各種機能を利用するプログラムを作成することができる.また QGIS にはプロセッシングフレームワークという各種データ加工アルゴリズムを組み合わせて利用する仕組みがある.最近の機能追加ではプロセッシングで用意されている各種アルゴリズムをコマンドラインから利用することができる QGIS Processing Executor という機能ができたのでバッチ処理を作成する時に利用できる[8].

GDAL は数多くのベクタ,ラスタ形式データを扱うための地理空間情報系データ用抽象ライブラリである[9].ここで言う抽象ライブラリとは多数の種類のデータについて同じような処理を行う抽象モデルを設定してそれらの処理を行う抽象関数を用意する仕組みである.地理空間系データの処理でいえば読み,書きから表示,座標変換,クリッピング等 GIS で利用されるようなデータ操作が抽象関数として用意されるとよい.このように用意されている抽象モデルを継承するデータ種類別のドライバクラスを用意し、それぞれのドライバで抽象関数を置き換える当該データ形式用実関数を用意する.プログラム実行時は抽象関数で呼ばれている箇所は対象データ形式の実関数が実行される.この方式で多数のデータに対応したデータ処理の仕組みを実現できる.たとえば GDAL のベクタデータのドライバには Shape, GeoPackage, PostGIS/PostgreSQL 等様々な形式のデータのものがある.これらはベクタデータとして GIS 上で利用する場合の処理は表示や検索,加工等大部分は同じものになる.ここに新たな形式のデータ利用を追加したい場合は GDAL のドライバインターフェースに沿った形でそのデータ用のドライバを書き GDAL パッケージに同梱すればよい.QGIS や ArcGIS は GDAL を利用しているため GDAL 用のドライバを用意できればそのデータ形式は QGIS や ArcGIS での利用が可能になる.アプリケーション側からはそのデータを扱うプログラム中の箇所を抽象モデルに対する抽象関数呼び出しで記述しておけば新しいデータ形式の扱いが追加されても GDAL の仕組みが対応するためアプリケーションプログラムの書き換えが不要である.GDAL はプログラムから利用する抽象モデルや関数以外にコマンドラインから利用できる多数の空間データ操作コマンドがある.データ形式の変換やクリッピング.タイル化等様々なコマンドがコマンドラインから利用できるのでバッチ処理を作成する場合に利用できる.また GDAL は QGIS プロセッシングプロバイダとして標準で登録されているのでプロセッシングから利用することも可能である.近年は GDAL 以外にも PDAL[10](点群データ用抽象ライブラリ) MDAL[11](メッシュデータ用抽象ライブラリ)等の空間情報系抽象ライブラリが整備されてきて QGIS 等で利用されている.

データ加工の結果については PostGIS/PostgreSQL のベクタデータとして出力することにした[12].

PostgreSQL はオープンソースのデータベース管理システムで様々な場所で利用されている.PostGIS は PostgreSQL 上で空間情報の利用ができるようにする拡張機能である.PostgreSQL のデータとしてベクタ,ラスタのデータを格納できるようにしたほかに各種空間演算を行う関数が用意され,PostgreSQL で用意されている SQL や API で利用することができる.

PostgreSQL には SQL として多彩な機能が用意されている[13].ここに適切な形でデータを格納すれば SQL を利用した様々な検索や加工にデータを利用することができる.また PostgreSQL はネットワーク中にサーバをたてて複数の端末からアクセスすることが可能である.設定されたプロトコルにしたがってアクセスできれば SQL 用クライアント(psql)だけではなく様々なプログラム（Excel, Access 等）から利用することが可能である.空間情報系でも GDAL や QGIS 等多数のプログラムが PostGIS/PostgreSQL に対応しているので空間データの表示,検索,編集,変換等の作業をネットワーク共用サーバを利用して行うことができる.

またデータ加工の途中のワークデータの形式としては GeoPackage をいくつかの場所で利用した.GeoPackage は OGC の標準になっているデータ形

式であり SQLite データベースでベクトルデータやラスタタイルが利用できる拡張を行ったものである[14].SQLite は単一ファイル内で複数のテーブルを作成,操作可能なデータベース管理システムである.PostgreSQL のような共用データベースとして利用することはできないが QGIS 等での操作で複数のデータソースを利用する場合は関連する複数のファイル(ベクタデータ,レイヤシンボロジ,プロジェクト定義）等を 1 個のファイルにまとめることができるので個別プロジェクトの管理や受け渡しを行う場合の利便性が高い.

2.6. QGIS の Processing 機能について

近年の QGIS では多くの空間データ加工の機能を Processing というフレームワークで提供している[15].

通常の Processing ではメニューからプロバイダとデータ加工アルゴリズムを指定すると表示されるダイアログでパラメータを指定してデータ加工を実行する.また同じ加工をしたいデータが複数ある場合はバッチ処理指定画面でパラメータを指定してバッチ処理を行うことができる.

Processing で言うプロバイダとは各種データ加工アルゴリズムをまとめて提供している提供元である.プロバイダには QGIS 以外にも GDAL, PDAL, GRASS, R, SAGA 等様々な外部提供の空間データ加工アルゴリズムがある.QGIS のプラグインとして既存プログラムを実行するためのパラメータを指定するプログラムを書けばプラグインの追加を行うことができる.また Processing の各アルゴリズムは Python のスクリプトから呼び出すことができる.そのため QGIS 用のプラグインを書く場合や QGIS の外部で起動する Python スクリプトを書くときに Processing のアルゴリズムを呼び出して利用することが可能である.

Processing では空間データ加工処理のために標準化された入力と出力のインターフェースを持ったデータ加工モジュール群である.入出力がいくつかの GIS 用データ向けに標準化されているため,提供されている複数のモジュールの入出力を結びあわせて連続処理を記述することができる.Processing には既存のデータ加工アルゴリズムを GUI で結び付けて新しいデータ加工モデルを作成するグラフィカルモデラーという機能がある.

また最近の機能追加で Processing で用意されている各アルゴリズムをコマンドラインから個別に実行できるようになった.今回のデータ加工ではバッチ処理として様々なデータ加工を行ったがこのコマンドラインからの Processing 機能も利用した.

図 1. Processing toolbox

3. データ加工作業と問題対応

今回のデータ加工については Shape 形式で集められた浸水想定区域データを都道府県単位で加工を行うプログラム群を用意して動作検証と再調整をおこなった.Shape 形式ファイルについては各浸水想定区域ポリゴンには想定浸水深のランクデータが格納されている.浸水想定区域のポリゴンは 5m メッシュで

提供されているものと同一浸水深ランクの連続領域ポリゴンでできているものがある.データは国土交通省の事務所,都道府県,市町村から集めてきたものである.サンプルとして作業を行った県のファイル数は約 2800 本である.

3.1. データ加工手順

データ加工の手順で最初の目標は各種浸水想定区域ポリゴンと 5m メッシュのオーバーレイを行い Union で作成された 5m メッシュのメッシュ番号と想定浸水深ランクを作業用のテーブルに格納することである.この時作業用のテーブルには同一 5m メッシュの ID が複数格納されていることを許可する(複数の浸水想定区域が同一の 5m メッシュと重なっている場合,単一の 5m メッシュがオーバーレイ処理で分割され複数のポリゴンができることを想定)

できあがった作業用テーブルから SQL でユニークな 5m メッシュ ID のリストを抽出して結果をテーブルにする.この時同一 ID のメッシュのレコードが作業用テーブルに複数存在する場合はそれらのレコードが持つ浸水想定深ランク値で一番深い浸水深を表すランク値を抽出結果の浸水深ランクの値にする.

ここでできあがったテーブルと 5m メッシュの地図図形つきテーブルをメッシュキーで結合かけると複数の浸水想定データを組み合わせた結果から一番浸水深ランク値が大きい値を抽出した 5m メッシュデータができる.これを利用すると想定浸水深ランク値を利用した色分け地図を描画することができる.

また 5m メッシュのデータレコードにその 5m メッシュが含まれている 4 次メッシュの ID が格納されているテーブルを作成して,そのテーブルをここで作成された作業テーブルと 5m メッシュ ID をキーにして結合させた結果のテーブルまたはビューを作成することができる.この結果テーブルで 4 次メッシュ番号をキーにして集計することによって 4 次メッシュ単位の浸水想定メッシュの数を算出することができる.この集計は想定浸水深ランク別に行うことも可能である.国勢調査をはじめとする各種統計データは 4 次メッシュ単位での集計値を公表しているものがたくさんある.今回の加工データの結果として 4 次メッシュ単位のデータを用意することによって地域の浸水想定領域のデータと各種統計データとを組み合わせた統計処理を行うことがやりやすくなる.

3.2. データ加工の問題点

データ加工は最初に Shape ファイルのリストが入ったテキストファイルを作成し最初の入力パラメータとした.Shape ファイルでは処理速度が遅いのですべての Shape ファイルを Geopackage に変換してファイル名も連番に変換したものを記述したテキストファイルを作成した.以後各データ加工プログラムは Geopackage の連番ファイル名が格納されたテキストファイルを入力として動作させる.

ここで作成した GeoPackege と 5m メッシュポリゴンのオーバーレイ処理を行ったところ入力ポリゴンにエラーがありオーバーレイ処理ができないものがいくつか存在した.ポリゴンデータの座標列が自己交差しているようなものがあるとこのようなエラーが発生する.このようなエラーを修正するために QGIS の Processing にある fixgeometry というアルゴリズムを利用する.このアルゴリズムでは修正後の出力ファイル名を形式指定で指定できるので Geopackage のファイル名を指定することでジオメトリエラーの修正とファイル形式の変換を行うことができる[16].今回はこの作業をコマンドラインから実行する引数を Python スクリプトで用意してスクリプト内から実行した.

また浸水想定区域ポリゴンと 5m メッシュポリゴンのオーバーレイ処理を行い Union で作成される中間データを作成する処理では対象データ量が多すぎて処理に長時間かかるものがいくつかあった.ものによっては 2 週間処理を行っても終わらないものがいくつかあった.

2 個のポリゴンレイヤのオーバーレイ処理を行う場合お互いのレイヤの各レコードの位置関係を比較しなければいけないのでレコード数が多いレイヤ間でオーバーレイ処理を行うととても時間がかかってしまう.双方のレイヤで空間インデックスを作成していれば全部のデータの比較を行う必要はないが,

データ量が多いとどうしても処理時間が多くかかってしまう.

この現象への対応として浸水想定区域ポリゴンと5m メッシュポリゴンについてメッシュで分割したデータを作成しそれぞれ分割後のデータを掛け合わせてポリゴンオーバーレイ処理を行った.今回のオーバーレイ作業で必要な結果はオーバーレイで重なる部分がある 5m メッシュの ID と想定浸水深ランク数値のリストであり,ID が重複したものがあることは前提であるのでこの方法が利用できる.

分割処理については当初は２次メッシュでの分割を行ったがデータ量が多い箇所についてはこの処理でも１日ぐらいかかったので３次メッシュでの分割に途中で切り替えた.3 次メッシュ単位にデータを分割した後の個別オーバーレイ処理は 10 分以内ぐらいで終了する.

3.3. 処理の修正

オーバーレイ処理を 3 次メッシュ単位で行うにあたり処理内容の変更を行った.1 本の浸水想定ポリゴンに対する変更後のオーバーレイ処理手順は以下の通りである.

1. 浸水想定区域ポリゴンと 3 次メッシュの Intersect を行い 3 次メッシュで分割された浸水想定区域ポリゴンを作成する.
2. 作成した 3 次メッシュ分割浸水想定区域を 3 次メッシュ別の GeoPackage ファイルに分割する.
3. 3 次メッシュ単位に分割した浸水想定区域ポリゴンと 3 次メッシュ別 5m メッシュのオーバーレイ処理（Union）を行い結果の 5m メッシュ ID＋浸水深ランクデータを浸水想定区域ポリゴンファイル×3 次メッシュの単位で CSV ファイルとして出力した.
4. できあがった CSV ファイルを 3 次メッシュ単位に作成した PostgreSQL のテーブルにロードした.この時各レコードは複数の同一 5m メッシュのものがテーブル内に存在することを許す.

3.4. 結果ビューの作成

5m メッシュの ID リストが格納された PostgreSQL のテーブルと複数の浸水想定区域とのオーバーレイ結果を集めた作業用テーブルを結合かけてビューを作成すると複数のオーバーレイ処理の結果としてのビューを作成できる.この時結合元のテーブルにメッシュの地図図形データが格納されていれば GIS 上で地図描画に利用できる.表結合を行う場合オーバーレイ処理結果部分に格納されている想定浸水深ランク数値については複数存在する可能性がある同一 ID5m メッシュレコードのうち最深浸水深を表す数値を格納することが要求されている.今回はランク数値が大きいもののほうが深いという前提でコードが設定されているので外部結合用 SQL で当該値を取得するところの条件に最大数値を取得するという記述を行うことによって要求を満たすことができる.

また外部結合元のテーブルの各レコードに当該 5m メッシュが含まれる 4 次メッシュのメッシュコードが含まれている場合結合後のビューを 4 次メッシュ単位に集計することができる.4 次メッシュ単位に浸水想定 5m メッシュの数や浸水深ランク別メッシュ数を算出することができる.これによって地域の想定浸水深領域の割合等を 4 次メッシュ単位のテーブルとして算出することができ,4 次メッシュ単位に作成してある国勢調査をはじめとする統計データと組み合わせて利用することができる.

リレーショナルデータベースにおいてビューとはなんらかの検索の結果を仮想的な表とする仕組みである.仮想的な表であるから通常はビューにアクセスすると検索が発行されるのでテーブルに対するアクセスよりは時間がかかる.今回はこの結果のビューを利用して QGIS 等で地図描画を行うこともデータ加工処理の後で想定しているためアクセス速度は速いほうがいい.PostgreSQL ではマテリアライズドビュー[17]という高速アクセス可能なビューの仕組みがあるので今回はそれを利用した.

3.5. 加工処理の再実行

一連のデータ加工作業が終了した後に一部データの差し替えや加除修正が発生する場合がある.この場合全部のデータの再作成ではなく影響があるデータのファイルリストが格納されたパラメータファイルを作成して,その部分のデータ再加工処理を行え

ばいい.ただしオーバーレイ結果の CSV 掃き出し処理以降は同一想定浸水領域ポリゴンの CSV ファイルの削除とその浸水領域ポリゴンが重なる 3 次メッシュの CSV アップロードテーブルの削除と CSV の再アップロード,結果ビューの再作成が必要になる.

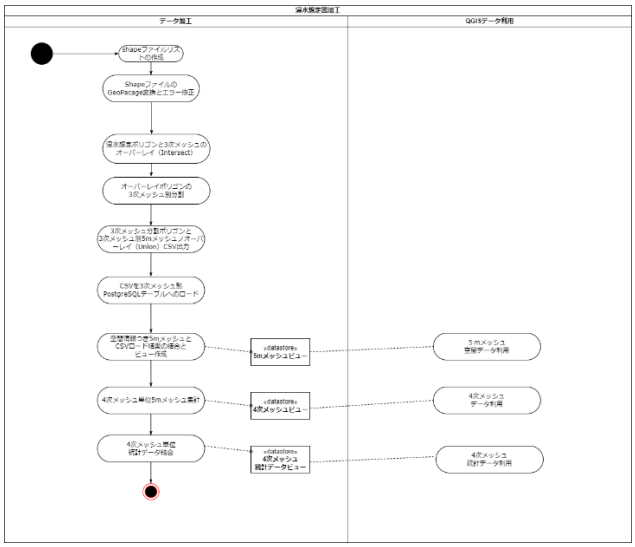

図 2.　データ加工手順

４．まとめ

近年の地理情報システムはグラフィカルユーザインターフェースを充実させてユーザがインタラクティブに操作を行う機能が充実してきている.しかし大量の空間情報データを加工する場合はコマンドラインからの処理を組み合わせたバッチ処理をどう組み立てられるかが重要である.バッチ処理にしてしまえば大量のファイル名称を入力パラメータとした処理の記述ができ,このようなパラメータは OS のディレクトリ操作コマンドとテキストエディタを操作すると作成することができる.

またバッチ処理での空間データ加工の手順を作成するためにはコマンドラインから利用できる空間データ加工プログラムやそのようなプログラムを作成することができる空間データ系ライブラリについての知見が重要になる.GDAL,PDAL,MDAL 等の空間情報系抽象ライブラリは空間情報の加工では重要である.また QGIS についてはデータ加工フレームワークの Processing についてコマンドラインからの利用や Python からの利用が提供されているので空間データ加工のためのバッチ処理記述のためには便利である.Processing では QGIS 等で用意されている多彩な空間データ加工手法が利用できるし QGIS 上で同一処理を動かすことが可能なのでデータ加工の試行と内容の検証を簡単に行うことができる.

PostGIS/PostgreSQL の利用は空間演算用関数を SQL で利用できることが重要である.SQL についてはデータベースの世界での利用実績が多いため高機能な実装がたくさんある.またデータベース管理システムを利用することで様々なデータとの連携やプログラムからの利用が可能になる/

大量データ間の空間演算はとても時間がかかる場合が多い.時間がかかる処理は失敗した場合もロスする時間が大きいので可能であるならばデータを小分けにして単一の処理にかかる時間を減らしたほうがいい.今回は浸水想定区域ポリゴンと 5m メッシュのオーバーレイ処理を 3 次メッシュ単位に分割することで単一処理にかかる時間を減少させることができたので処理終了の目途をたてることができた.

謝辞

本稿に記述した作業を行うにあたり日本放送協会大石寛人様, 浅野将様にご指示, データ準備と調整, ご助言いただきました. この場を借りて深く御礼申し上げます.

参考文献

[1]国土交通省　(2019) 洪水浸水想定区域図・洪水ハザードマップ. https://www.mlit.go.jp/river/bousai/main/saigai/tisiki/syozaiti/,(参照 2022-08-19)

[2]国土交通省　(2020) 不動産取引時において、水害ハザードマップにおける対象物件の所在地の説明を義務化 ～宅地建物取引業法施行規則の一部を改正する命令の公布等について～. https://www.mlit.go.jp/report/press/totikensangyo16_hh_000205.html, (参照 2022-08-19)

[3] 朝日新聞 (2018) ハザードマップと重なった浸水域、それでも犠牲者防げず,

https://www.asahi.com/articles/ASL7956K2L79PTIL02N.html. (参照 2022-08-19)
[4]日本放送協会 (2022) NHK 全国ハザードマップ Risk Map in Japan. https://www.nhk.or.jp/campaign/w-hazardmap/,(参照 2022-08-19)
[4]日本放送協会 (2022) 34 テラバイトのデータと格闘して「全国ハザードマップ」を公開した理由. https://www3.nhk.or.jp/news/special/saigai/select-news/20220621_01.html (参照 2022-08-19)
[5] 総務省統計局, 地域メッシュ統計の特質・沿革. https://www.stat.go.jp/data/mesh/pdf/gaiyo1.pdf (参照 2022-08-19)
[6] 国土地理院, 高精度な標高データ. https://www.gsi.go.jp/kankyochiri/Laser_demimage.html (参照 2022-08-19)
[7] OSGeo4W FOSSGIS for Windows. https://trac.osgeo.org/osgeo4w/wiki/WikiStart (参照 2022-08-20)
[8] QGIS user manual, 24.8. プロセシングをコマンドラインから使用する. https://docs.qgis.org/3.22/ja/docs/user_manual/processing/standalone.html (参照 2022-08-20)
[9] GDAL documentation. https://gdal.org/ (参照 2022-08-20)
[10] PDAL - Point Data Abstraction Library. https://pdal.io/en/stable/ (参照 2022-08-20)
[11] MDAL. https://www.mdal.xyz/ (参照 2022-08-20)
[12] About PostGIS. https://postgis.net/ (参照 2022-08-20)
[13] PostgreSQL 14.0 文書 パートⅡ. SQL 言語. https://www.postgresql.jp/document/14/html/sql.html (参照 2022-08-20)
[14] GeoPackage. https://www.geopackage.org/ (参照 2022-08-20)
[15] 嘉山陽一 (2020). QGIS プロセッシング利用のすすめ. https://www.aeroasahi.co.jp/qgis/post/2020/12/processing_202012/ (参照 2022-08-21)
[16] ジオメトリの修復. https://docs.qgis.org/3.22/ja/docs/user_manual/processing_algs/qgis/vectorgeometry.html#fix-geometries (参照 2022-08-21)
[17] PostgreSQL 14.0 文書. マテリアライズドビュー. https://www.postgresql.jp/document/14/html/rules-materializedviews.html (参照 2022-08-21)