
PostgreSQL エンタープライズ・コンソーシアム 技術部会 WG#2

GIS アセスメント編

製作者
担当企業名
株式会社富士通ソーシャルサイエンスラボトリ

改訂履歴

版	改訂日	変更内容
1.0	2016/4/11	新規作成

ライセンス



本作品は CC-BY ライセンスによって許諾されています。

ライセンスの内容を知りたい方は <http://creativecommons.org/licenses/by/2.1/jp/> でご確認ください。

文書の内容、表記に関する誤り、ご要望、感想等につきましては、PGECons のサイトを通じてお寄せいただきますようお願いいたします。

サイト URL <https://www.pgecons.org/contact/>

Eclipse は、Eclipse Foundation Inc の米国、およびその他の国における商標もしくは登録商標です。

IBM および DB2 は、世界の多くの国で登録された International Business Machines Corporation の商標です。

Intel、インテルおよび Xeon は、米国およびその他の国における Intel Corporation の商標です。

Java は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

Linux は、Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。

Red Hat および Shadowman logo は、米国およびその他の国における Red Hat, Inc. の商標または登録商標です。

Microsoft、Windows Server、SQL Server、米国 Microsoft Corporation の米国及びその他の国における登録商標または商標です。

MySQL は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

Oracle は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

PostgreSQL は、PostgreSQL Community Association of Canada のカナダにおける登録商標およびその他の国における商標です。

Windows は米国 Microsoft Corporation の米国およびその他の国における登録商標です。

TPC、TPC Benchmark、TPC-C、TPC-E、tpmC、TPC-H、QphH は米国 Transaction Processing Performance Council の商標です。

その他、本資料に記載されている社名及び商品名はそれぞれ各社が商標または登録商標として使用している場合があります。

はじめに

■本資料の概要と目的

本資料は地理情報システムに求められる要件を PostGIS を用いてどのように応えるかを考察した文書です。

■資料内の記述について

本資料では下記フローに従い、地理情報データの分析を記載していきます。「はじめに」以降の各章に関しては、下記フローの項番名称に従い記述します。

本資料では各工程における、検討事項や基本的なツールによるコマンドの例示を記載します。

なお、PostGIS ならびに pgRouting のインストール手順については本書では省略しています。

別紙「PostGIS インストール手順書」及び「pgRouting インストール手順書」を参照して下さい。



図 1:本資料の流れ

■本資料で扱う用語の定義

本資料では曖昧な意味として捉えられる用語を用います。各用語は次のような意味で記載しています。

表 1:用語定義

No.	用語	意味
1	GIS (じーあいえす)	GIS(Geographic Information System/地理情報システム)とは地理情報(位置に関連付けられた情報)を要件に応じて管理、加工し、検索結果や分析結果を出力(可視化)するシステムを指します。
2	DBMS (でーびーえむえす)	データベース管理システムを指します。本章では、PostgreSQL の総称として利用します。
3	オブジェクト	PostgreSQL が取り扱うデータ型や DBMS 変数等の総称として用います。
4	空間データベース	文字型や数値型のデータ同じように、地理情報のデータを SQL を用いて処理できるデータベース(DBMS)を指します。

■本資料で扱うソフトウェア

本資料では、次のソフトウェアを用いてコマンド例証等を挙げています。

表 2:動作環境

No.	環境名	実装	バージョン
1	検証対象 DBMS	PostgreSQL	9.5.1
2	検証対象空間データベース	PostGIS	2.2.1
3	検証対象オペレーティングシステム	CentOS	intel CPU CentOS release 6.7 (Final)
4	デスクトップ型 GIS ソフトウェア	QGIS	2.14.0

表 3:コマンド・ツール一覧

No.	コマンド・ツール	バージョン	ライセンス	入手元(URI)	概要
1	COPY (こぴー)	— (PostgreSQL 標準コマンド)	The PostgreSQL License	(PostgreSQL 同梱)	PostgreSQL においてファイルとテーブル間のデータをコピーするためのコマンドです。本書では、データが列挙されているファイルからテーブルへデータを挿入する際に利用します。
2	psql (びー・えす・きゅー・える)	9.5.1	The PostgreSQL License	(PostgreSQL 同梱)	PostgreSQL に同梱されている、ターミナル型フロントエンドです。ファイルから入力を読み込むことも可能です。
3	PostGIS (ぽすとじす)	2.2.1	GNU General Public License, version2	http://postgis.net/	PostgreSQL で地理空間情報を扱えるようにする機能拡張モジュールです。
4	shp2pgsql (しえーぶ・とう・ぴー・じー・えすきゅーえる)	2.2.1	GNU General Public License, version2	(PostGIS 同梱)	PostGIS に同梱されている、ESRI Shapefile 形式のファイルを PostgreSQL にインポート可能な SQL に変換するコマンドです。
5	QGIS (きゅー・じーあいえす)	2.14.0	GNU General Public License, version2	http://www.qgis.org/ja/site/	地理情報システムの閲覧、編集、分析機能を有するソフトウェアです。本書では、PostGIS に格納したデータを可視化するために利用します。
6	pgRouting (びー・じー・るーていんぐ)	2.0.0	GNU General Public License, version2	https://github.com/pgRouting/pgrouting/releases	PostGIS に経路検索機能を追加する機能拡張モジュールです。

目次

1. 事前検討.....	6
1.1. 地理情報データの選定.....	6
2. 地理情報データ収集.....	7
2.1. 地理情報データの公開サイト.....	7
2.2. 地理情報データのフォーマット.....	8
2.3. 要件を満たす地理情報データの取得.....	8
3. 地理情報データ変換.....	12
3.1. ESRI Shapefile の変換.....	12
3.2. shp2pgsql による ESRI Shapefile の変換.....	13
4. 地理情報データインポート.....	17
4.1. インポート先のデータベース作成.....	17
4.2. データインポート.....	20
4.3. 地理情報データのインポート確認.....	22
5. 地理情報データ分析.....	25
5.1. 要件の確認.....	25
5.2. 文化施設の多い行政区域の調査.....	25
5.3. 文化施設までの経路検索.....	35

1. 事前検討

旅行会社 A では、日本の文化・芸術等に興味を持つ海外旅行者のため、関東近郊(東京都および神奈川県)の文化施設を、短期間で訪れる新しい旅行プランを検討しています。

短期間の中で、より多くの文化施設を訪問するため、文化施設の多い地域、および、最適な観光ルートを検討するため、文化施設間の最適な経路を調査することにしました。

本書では旅行会社 A で新しい旅行プランを企画するために必要であった、下表の 2 つの要件を PostGIS および QGIS を用いて、実現するまでの手順を記載します。

表 1.1:GIS で実現したい要件

No.	要件	概要
1	文化施設の多い市町村の確認	市町村毎(行政区間毎)の文化施設数を集計し、地図上に表示します。
2	文化施設までの経路検索	ある地点から任意の文化施設までの経路を検索し、地図上に表示します。

1.1. 地理情報データの選定

上記に記載した 2 つの要件を満たすため、どのような地理情報データが必要か検討した結果、下表のデータが必要であると判断しました。

表 1.1.1:要件を満たす為に必要な地理情報データ

No.	地域	必要な情報	概要
1	東京都	行政区域情報	都道府県や市町村と言った区画情報
2	神奈川県		
3	東京都	文化施設情報	文化施設の位置情報
4	神奈川県		
5	東京都	道路情報	東京都および神奈川県の道路情報
6	神奈川県		

2. 地理情報データ収集

本章では公開された地理情報データの収集方法を記載した後、「1.1 地理情報データの選定」に記載したデータを取得する手順を記載します。

2.1. 地理情報データの公開サイト

下表に代表的な、地理情報データの公開サイトの一部を記載します。日本国では各省庁のサイトから要件に様々な応じた地理情報データを検索し、ダウンロードする事が可能です。

なお、地理情報データファイルのフォーマットに関しては、「2.2 地理情報データのフォーマット」に記載します。

表 2.1.1: 代表的な空間地理データ公開サイト

No.	提供機関/サイト名	URL	ファイルフォーマット	主なデータ項目
1	国土地理院/ 基盤地図情報 ダウンロードサービス	http://fgd.gsi.go.jp/download/	JPGIS(GML)	測量の基準点、海岸線、行政区画の境界線及び代表点、道路縁、軌道の中心線、標高点等
2	国土交通省/ 国土数値情報 ダウンロードサービス	http://nlftp.mlit.go.jp/ksj/	JPGIS(GML)、Shapefile	国土、行政区画、地域、交通、各種統計等
3	環境省/ 自然環境調査 GIS	http://gis.biodic.go.jp/webgis/	Shapefile、KML	植生調査、特定植物群調査、巨樹・巨木林調査、河川調査、海岸改善状況調査等

2.2. 地理情報データのフォーマット

地理情報データの代表的なファイルフォーマットを下表に記載します。

表 2.2.1: 地理情報データの代表的なファイル形式

No.	フォーマット	概要	データ形式
1	ESRI Shapefile (Shapefile)	ArcGIS ¹ という商用ソフトウェアを提供している ESRI 社が定義したフォーマット。現在 GIS で最も利用されているファイルフォーマットです。Shapefile は名前が同一で、拡張子(.shp,.shx,.dbf,prj 等)が異なるいくつかのファイルで構成されています。(「2.2.1 ESRI Shapefile の構成」に Shapefile を構成するファイルを記載します。) Shapefile はバイナリ形式でデータを保持しているため、テキストエディタ等で編集することはできません。Shapefile に対応した専用のソフトウェアが必要です。	ベクタ形式
2	GeoJSON	JSON(JavaScript Object Notation)を元にした地理情報データのファイルフォーマット GeoJSON の仕様については下記 URL をご参照下さい。 http://geojson.org/geojson-spec.html	ベクタ形式
3	CSV	値をカンマ(,)で各列を区切ったテキストファイルです。1行目にカラム名とし、2行目以降に実際のデータを記録されたものが多いです。注意点としてポイントデータのみで、ラインやポリゴンを扱う事は出来ません。似た形式として、「スペース区切り」、「タブ区切り」等のファイルフォーマットが存在します。	ベクタ形式
4	XML	汎用的なマークアップ言語として策定された XML (Extensible Markup Language) で地理情報データを表現するフォーマット。XML に準拠したファイルフォーマットとしては、GML (Geography Markup Language)、KML (Keyhole Markup Language) がよく利用されています。	ベクタ形式
5	GML	Open Geospatial Consortium (OGC)によって開発された地理情報データを表現するための XML ベースのマークアップ言語です。	ベクタ形式
6	KML	3次元の地理情報データを表現するために開発された XML ベースのマークアップ言語。Google Earth として知られているソフトウェアの旧名「Keyhole」で利用されていたファイルフォーマット「Keyhole Markup Language」の頭文字が名前の由来となっている。現在も、Google Earth で利用されています。	ベクタ形式
7	JPGIS(Japan Profile for Geographic Information Standards)	最新の地理情報に関する国際規格 (ISO19111シリーズ)、日本工業規格 (JISX7111シリーズ)に準拠し、内容を整理した日本独自の標準規格です。日本の独自規格ですが、行政組織が配布しているデータは JPGIS 形式のものが多くあります。	ベクタ形式

2.2.1. ESRI Shapefile の構成

ESRI Shapefile は、GIS で最も利用されているファイルとして挙げられます。

拡張子が異なる同名の複数ファイル群で構成されます。ESRI Shapefile を構成するファイル群を下表に示します。

表 2.2.1.1: ESRI Shapefile のファイル構成

No.	ファイル拡張子	概要
1	dbf ファイル	属性情報を記載したファイル
2	prj ファイル	Shapefile をどのような投影法や地球楕円体に基づいた空間参照情報で表示するかを記載した定義ファイル
3	shp ファイル	図形の座標情報を記載したファイル
4	shx ファイル	shp の図形と dbf の属性の対応関係を記載したファイル

¹ <http://www.esri.com/products/arcgis/>

2.3. 要件を満たす地理情報データの取得

前述の地理情報データを公開するサイトを調査した結果、「国土数値情報(<http://nlftp.mlit.go.jp/ksj/>)」から、本書の要件を満たすための下表の地理情報データを取得可能なことが確認できました。

表 2.3.1:要件を満たすために必要なファイル一覧

No.	概要	地域	測地系	ダウンロードしたファイル名
1	行政区域データ	東京都	世界測地系	N03-150101_13_GML.zip
2	行政区域データ	神奈川県	世界測地系	N03-150101_14_GML.zip
3	道路データ	東京	世界測地系	N01-07L-13-01.0a_GML.zip
4	道路データ	神奈川県	世界測地系	N01-07L-14-01.0a_GML.zip
5	文化施設データ	東京都	世界測地系	P27-13_13.zip
6	文化施設データ	神奈川県	世界測地系	P27-13_14.zip

以下に参考として東京都および神奈川県の「行政区域データ」をダウンロードする手順を記載します。同様の手順で、東京都および神奈川県の「道路データ」および「文化施設データ」もダウンロード可能です。「道路データ」および「文化施設データ」のダウンロード手順は省略します。）

国土数値情報のウェブサイト(<http://nlftp.mlit.go.jp/ksj/>)にアクセスし、表示されたテーブル上のセル「データ形式」から「GML(JPGIS 2.1) シェープファイル」セルを押下、次いで「政策区域」テーブル配下の「行政区域」セルを押下すると、「データのダウンロード(2.各データ詳細)」画面に遷移します。該当のページには「データの内容」、「座標系(SRID)」、「データ形状」、「データ構造」、「データイメージ」等、ダウンロードするデータに関わる詳細が記載されています。

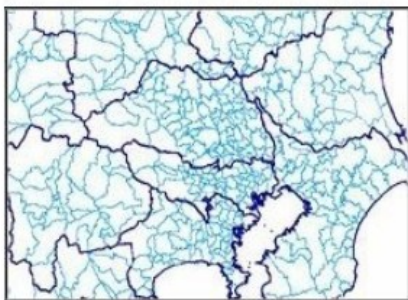
座標系	JGD2000 / (B, L)	
データ形状	面	
データ構造	<div> <div> <div><<Feature>></div> <div>行政区域境界</div> <div>+範囲:GM_surface</div> <div>+都道府県名:CharacterString</div> <div>+支庁名[0..1]:CharacterString</div> <div>+都・政令都市名:CharacterString</div> <div>+行政区域コード:行政コード</div> </div> <div> <div><<CodeList>></div> <div>行政コード</div> <div>(from共通パッケージ)</div> <div>+都道府県コード:都道府県コード</div> <div>+市町村コード:市町村コード</div> </div> </div>	
データイメージ	<div> <div>《拡大表示するには図をクリックしてください》</div>  </div>	

図 2.3.1:データ詳細抜粋

本画面を下にスクロールし、「ダウンロードするデータの選択」テーブルから「東京都」と「神奈川県」を選択し、「次へ」ボタンを押下します。

ダウンロードするデータの選択							
<input type="checkbox"/> 全国							
<input type="checkbox"/> 北海道	<input type="checkbox"/> 青森	<input type="checkbox"/> 岩手	<input type="checkbox"/> 宮城	<input type="checkbox"/> 秋田	<input type="checkbox"/> 山形	<input type="checkbox"/> 福島	<input type="checkbox"/> 茨城
<input type="checkbox"/> 栃木	<input type="checkbox"/> 群馬	<input type="checkbox"/> 埼玉	<input type="checkbox"/> 千葉	<input checked="" type="checkbox"/> 東京	<input checked="" type="checkbox"/> 神奈川	<input type="checkbox"/> 新潟	<input type="checkbox"/> 富山
<input type="checkbox"/> 石川	<input type="checkbox"/> 福井	<input type="checkbox"/> 山梨	<input type="checkbox"/> 長野	<input type="checkbox"/> 岐阜	<input type="checkbox"/> 静岡	<input type="checkbox"/> 愛知	<input type="checkbox"/> 三重
<input type="checkbox"/> 滋賀	<input type="checkbox"/> 京都	<input type="checkbox"/> 大阪	<input type="checkbox"/> 兵庫	<input type="checkbox"/> 奈良	<input type="checkbox"/> 和歌山	<input type="checkbox"/> 鳥取	<input type="checkbox"/> 島根
<input type="checkbox"/> 岡山	<input type="checkbox"/> 広島	<input type="checkbox"/> 山口	<input type="checkbox"/> 徳島	<input type="checkbox"/> 香川	<input type="checkbox"/> 愛媛	<input type="checkbox"/> 高知	<input type="checkbox"/> 福岡
<input type="checkbox"/> 佐賀	<input type="checkbox"/> 長崎	<input type="checkbox"/> 熊本	<input type="checkbox"/> 大分	<input type="checkbox"/> 宮崎	<input type="checkbox"/> 鹿児島	<input type="checkbox"/> 沖縄	

図 2.3.2:都道府県を選択

年代の選択に移しますので、任意の年代を選択(本書では直近の年代)し、「次へ」ボタンを押下します。下図では、平成 27 年の東京都および神奈川県のデータを選択しています。

<input type="checkbox"/> N03-130401_13_GML.zip	3.86MB	平成25年	世界測地系	東京
<input type="checkbox"/> N03-140401_13_GML.zip	3.95MB	平成26年	世界測地系	東京
<input checked="" type="checkbox"/> N03-150101_13_GML.zip	7.26MB	平成27年	世界測地系	東京

図 2.3.3:平成 27 年度データの選択(東京都)

<input type="checkbox"/> N03-140401_14_GML.zip	1.81MB	平成26年	世界測地系	神奈川
<input checked="" type="checkbox"/> N03-150101_14_GML.zip	3.06MB	平成27年	世界測地系	神奈川

図 2.3.4:平成 27 年度データの選択(神奈川県)

最後に「アンケートのご協力をお願い」画面に移ります。アンケートに回答し、「国土数値情報利用約款」を確認し理解した上で「はい」ボタンを押下し、データをダウンロードします。

上記の手順でダウンロードした東京都および神奈川県の「」、「道路データ」の仕様を確認したところ、下表の地理情報データあることが確認できました。

表 2.3.2:地理情報データの諸元

No.	概要	地域	ファイル名	測地系(SRID)	地理情報データ
1	行政区域データ	東京都	N03-150101_13_GML.zip	JGD2000(4612)	ポリゴン
2	行政区域データ	神奈川県	N03-150101_14_GML.zip	JGD2000(4612)	ポリゴン
3	道路データ	東京	N01-07L-13-01.0a_GML.zip	JGD2000(4612)	ライン
4	道路データ	神奈川県	N01-07L-14-01.0a_GML.zip	JGD2000(4612)	ライン
6	文化施設データ	東京都	P27-13_13.zip	JGD2000(4612)	ポイント
7	文化施設データ	神奈川県	P27-13_14.zip	JGD2000(4612)	ポイント

ダウンロードした SRID 4612 の地理情報データを PostGIS で処理可能か否かを確認するため、PostGIS を導入したデータベースにログインし、「spatial_ref_sys」テーブルを参照する SQL を実行しました。

「spatial_ref_sys」テーブルには、PostGIS の拡張が行われたデータベースで扱える SRID の情報が格納されています。実行結果の通り、SRID 4612 の地理情報データを PostGIS で処理可能なことを確認できました。

```
$ psql
=# \x
Expanded display is on.
=# SELECT * FROM spatial_ref_sys WHERE srid = '4612';
-[ RECORD
1 ]-----
srid      | 4612
auth_name | EPSG
auth_srid | 4612
srtxt     | GEOGCS["JGD2000", DATUM["Japanese_Geodetic_Datum_2000", SPHEROID["GRS
1980", 6378137, 298.257222101, AUTHORITY["EPSG", "7019"]], TOWGS84[0, 0, 0, 0, 0, 0], AUTHORITY["EPSG", "6612"]]
, PRIMEM["Greenwich", 0, AUTHORITY["EPSG", "8901"]], UNIT["degree", 0.0174532925199433, AUTHORITY["EPSG", "912
2"]], AUTHORITY["EPSG", "4612"]]
proj4text | +proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0 +no_defs
```

図 2.3.4: PostGIS で利用可能な SRID か否かを確認

3. 地理情報データ変換

本章では ESRI Shapefile を PostgreSQL にインポート可能な形式にするための方法を記載します。

3.1. ESRI Shapefile の変換

PostgreSQL では ESRI Shapefile をそのままインポートすることができないため、ESRI Shapefile のデータを PostgreSQL にインポート可能な形式(SQL ファイル)に変換する必要があります。このため PostGIS に付属する「shp2pgsql」ツールを用いて、ESRI Shapefile を SQL ファイルに変換します。

「shp2pgsql」の主な機能を下表に記載します。

表 3.1.1: shp2pgsql の機能

No.	機能	詳細
1	データの変換	Shapefile を一般的な SQL、又は PostgreSQL のダンプファイルに変換します。
2	DDL 文と DML 文の分離	一つの Shapefile からテーブル作成用の DDL とデータ投入用の DDL を作成可能です。これによって複数のシェープファイルに別れたデータを容易にインポートします。

東京都の行政区域データ(N03-150101_13_GML.zip)を例に用いて ESRI Shapefile を SQL ファイルに変換する手順を記載します。「N03-150101_13_GML.zip」には下表のファイルが格納されています。

表 3.1.2: N03-150101_13_GML.zip 格納ファイル

No.	ファイル名	概要
1	KS-META-N03-15_13_150101.xml	東京都の行政区域データのメタ情報を記載したファイルです。
2	N03-15_13_150101.dbf	shp ファイル内にある図形の属性情報を記載したファイルです。
3	N03-15_13_150101.prj	Shapefile をどのような投影法や地球楕円体に基づく空間参照情報で表示するかを記載した定義ファイルです。
4	N03-15_13_150101.shp	図形の座標情報を記載したファイルです。
5	N03-15_13_150101.shx	shp の図形と dbf の属性の対応関係を記載したファイルです。
6	N03-15_13_150101.xml	JPGIS のデータ本体。本データを国土数値情報が配布するデータ変換ツールで変換すると Shapefile を生成する事も可能です。本手順書では省略します。

東京都の行政区域データ(ESRI Shapefile)を PostgreSQL にインポート可能な形式に変換するコマンドを示します。

```
$ shp2pgsql -s 4612 -W cp932 -D -I -i N03-15_13_150101.shp maps.gyosei_kuiki > gyosei_kuiki_tokyo.dmp
Shapefile type: Polygon
Postgis type: MULTIPOLYGON[2]
$ ls -l gyosei_kuiki_tokyo.dmp
-rw-rw-r--. 1 postgres postgres 15057395 3月 15 14:53 gyosei_kuiki_tokyo.dmp
```

図 3.1.1: shp2pgsql による Shapefile の変換

3.1.1. 変換時の指定オプション

shp2pgsql で Shapefile を SQL ファイルに変換した際に使用したオプションは下表の通りです。

表 3.1.1.1: shp2pgsql のオプション

No.	オプション名	詳細
1	-s 4612	変換するデータの SRID(空間参照系)が 4612(JGD2000 地理座標系)である事を明示
2	-W cp932	入力ファイルのエンコーディングを cp932(Shift_JIS を拡張したエンコード)に指定
3	-D	PostgreSQL のダンプファイル形式による出力を指定
4	-I	geocolumn に INDEX を作成
5	-i	全ての整数を 32 ビット整数(integer 型)に変換
6	N03-15_13_150101.shp	変換対象のデータ(ESRI Shapefile)

7	maps.gyosei_kuiki	スキーマ名.テーブル名を指定
8	> gyosei_kuiki_tokyo.dmp	変換結果が標準出力に出力されるのでリダイレクトしてファイルに保存

3.1.2. 変換で出力されたファイル

shp2pgsql は ESRI Shapefile を PostgreSQL に対応した SQL 文に変換します。出力されたファイル (gyosei_kuiki_tokyo.dmp) の DDL 部分を下表に記載します

表 3.1.2.1: shp2pgsql の変換結果(DDL 部分を抜粋)

No.	コマンド	概要
1	SET CLIENT_ENCODING TO UTF8;	エンコーディングの設定
2	SET STANDARD_CONFORMING_STRINGS TO ON;	バックスラッシュをエスケープしない
3	BEGIN; CREATE TABLE "maps"."gyosei_kuiki" (gid serial, "n03_001" varchar(10), "n03_002" varchar(20), "n03_003" varchar(20), "n03_004" varchar(20), "n03_007" varchar(5));	テーブルを作成
4	ALTER TABLE "maps"."gyosei_kuiki" ADD PRIMARY KEY (gid);	インデックスを作成
5	SELECT AddGeometryColumn('maps', 'gyosei_kuiki', 'geom', '4612', 'MULTIPOLYGON', 2);	ジオメトリカラムを 既存のテーブルに追加

AddGeometryColumn 関数はジオメトリカラムを既存の属性テーブルに追加する関数です。
 No.5 の「AddGeometryColumn」関数は下表の様な値を与えられています。

```
SELECT AddGeometryColumn(① 'maps', ② 'gyosei_kuiki', ③ 'geom', ④ '4612', ⑤ 'MULTIPOLYGON', ⑥ 2);
```

図 3.1.2.1: AddGeometryColumn 部分抜粋

表 3.1.2.2: AddGeometryColumn の引数

No.	概要	引数の型	詳細
①	スキーマ名	文字列型	カラムを追加したいテーブルが存在するスキーマ名
②	テーブル名	文字列型	カラムを追加したいテーブル名
③	カラム名	文字列型	新規に追加したいカラム名
④	SRID (空間参照系)	数値型	データの SRID (空間参照系)を指定
⑤	ジオメトリ型	文字列型	格納された値がどのようなジオメトリ型を根拠とした値であるかを指定
⑥	次元	数値型	2 次元または 3 次元を指定

上記 DDL 文で作成した gyosei_kuiki テーブルは下表の列を持つテーブルです。

表 3.1.2.3: gyosei_kuiki テーブル

No.	列名	データ型	格納データ
1	gid	serial	レコードを一意に特定するための ID
2	n03_001	varchar (10)	都道府県名。当該区域を含む都道府県名称
3	n03_002	varchar (20)	支庁・振興局名。〇〇振興局、〇〇県民局等の事務所名
4	n03_003	varchar (20)	郡・政令都市名。当該行政区の郡又は政令市の名称
5	n03_004	varchar (20)	市町村名。当該行政区の市区町村の名称
6	n03_007	varchar (5)	行政区画コード
7	geom	MULTIPOLYGON	行政区画の形状を表すマルチポリゴン

出力されたファイルの一部(DML 部分)を下表に記載します。(1行を抜粋)

表 3.1.2.4: shp2pgsql の変換結果(DML 部分)

No.	コマンド	概要
1	<code>COPY "maps"."gyosei_kuiki" ("n03_001","n03_002","n03_003","n03_004" ,"n03_007",geom) FROM stdin;</code>	gyosei_kuiki テーブルにデータを COPY
2	東京都	n03_001(都道府県名)カラムに COPY されるデータ
3	¥N	n03_002(支庁名)カラムに COPY されるデータ
4	江東区	n03_003(郡・政令都市名)カラムに COPY されるデータ
5	¥N	n03_004(市町村名)カラムに COPY されるデータ
6	13108	n03_007(行政区域コード)カラムに COPY されるデータ
7	0106000020041200000100000001030000000100 000005000000192D65EBB97861400E1D5244BA CE41402B772064B9786140E2A581DDB8CE414 0AE396DB1AE78614056490E1FEBCE41405EDA A349AF786140113A20BDECCE4140192D65EBB 97861400E1D5244BACE4140	geom(市町村区画の地理情報データ)カラムに COPY されるデータ

下图の様に「-a」オプションを与えることで DML 文(COPY)だけを出力する事も可能です。

```
$ shp2pgsql -s 4612 -W cp932 -D -a -i N03-15_14_150101.shp maps.gyosei_kuiki > gyosei_kuiki_kanagawa.dmp
Shapefile type: Polygon
Postgis type: MULTIPOLYGON[2]
```

図 3.1.2.2: DML 文だけのダンプファイルに変換

shp2pgsql 下表のオプションを与えることで任意の DDL 文を追加する事が可能です。DDL 文を追加するオプションは DML だけを出力する「-a」オプションと排他的な関係にあります。

表 3.1.2.5: 出力データ選択オプション

No.	オプション名	概要
1	-d	ダンプファイルに DROP TABLE を含めます。
2	-c	ダンプファイルに CREATE TABLE を含めます。(オプションを指定しない場合のデフォルトの動作)
3	-p	ダンプファイルに CREATE TABLE 文だけを出力します。

以上で東京都の行政区域データの変換は完了です。

上記の東京都の行政区画データの変換と同様の手順で、他のデータに関しても変換を実施して下さい。本書では、下記コマンドで各データの変換を実施しました。

表 3.1.2.6: データ変換コマンド

No.	データ概要	都道府県	データ変換コマンド
1	行政区画データ	東京都	\$ shp2pgsql -s 4612 -W cp932 -D -I -i N03-15_13_150101.shp maps.gyosei_kuiki > gyosei_kuiki_tokyo.dmp
2		神奈川県	\$ shp2pgsql -s 4612 -W cp932 -D -a -i N03-15_14_150101.shp maps.gyosei_kuiki > gyosei_kuiki_kanagawa.dmp
3	道路データ	東京	\$ shp2pgsql -s 4612 -W cp932 -D -I -i N01-07L-2K-13_Road.shp maps.road_map > road_map_tokyo.dmp
4		神奈川県	\$ shp2pgsql -s 4612 -W cp932 -D -a -i N01-07L-2K-14_Road.shp maps.road_map > road_map_kanagawa.dmp
5	文化施設データ	東京都	\$ shp2pgsql -s 4612 -W cp932 -D -I -i P27-13_13.shp maps.bunkasetu > bunkasetu_tokyo.dmp
6		神奈川県	\$ shp2pgsql -s 4612 -W cp932 -D -a -i P27-13_14.shp maps.bunkasetu > bunkasetu_kanagawa.dmp

4. 地理情報データインポート

本章では shp2pgsql を用いて変換したデータを PostgreSQL のデータベースにインポートするまでの手順を記載します。

4.1. インポート先データベース作成

データをインポートするには、PostGIS の機能が有効化された、地理空間情報に対応した空間データベースが必要です。本節では PostgreSQL に地図用のデータベースを作成し、PostGIS を有効化するまでを記載します。

本書では下表の値でデータベースを作成します。

表 4.1.1 作成するデータベース

No.	項目	詳細
1	データベースユーザ名/パスワード	maps/maps
2	データベース名	maps
3	スキーマ名	maps
4	行政区域テーブル名	gyosei_kuiki
5	文化施設テーブル名	bunkasetu

4.1.1. データベースユーザ作成

地理情報データを扱う一般ユーザを作成します。

```
$ psql
=# CREATE USER maps WITH LOGIN PASSWORD 'maps';
CREATE ROLE
=# \du
```

Role name	Attributes	Member of
maps		{}
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	{}

(2 rows)

図 4.1.1.1: ユーザ作成

4.1.2. データベース作成

地理空間情報に対応したデータベースを作成します。

```
=# CREATE DATABASE maps OWNER maps TEMPLATE = template0 ENCODING = UTF8;
CREATE DATABASE
=# \l
```

Name	Owner	Encoding	Collate	Ctype	Access privileges
maps	maps	UTF8	C	C	
postgres	postgres	UTF8	C	C	
template0	postgres	UTF8	C	C	=c/postgres + postgres=Ctc/postgres
template1	postgres	UTF8	C	C	=c/postgres + postgres=Ctc/postgres

(4 rows)

図 4.1.2.1: データベース作成

4.1.3. スキーマ作成

地理空間情報用のテーブルを作成するためのスキーマを作成します。

```

¥q
$ psql maps -U maps
=> CREATE SCHEMA maps;
CREATE SCHEMA
=> ¥dn
List of schemas
Name | Owner
-----+-----
maps | maps
public | postgres
(2 rows)
=> ¥q

```

図 4.1.3.1: スキーマ作成

4.1.4. PostGIS の有効化

作成した maps データベースにて地理空間情報を扱えるように PostGIS を有効化します。有効化するには管理者権限が必要です。

```

$ psql maps -U postgres
=# CREATE EXTENSION postgis WITH SCHEMA maps;
CREATE EXTENSION
=# ¥dx
List of installed extensions
Name | Version | Schema | Description
-----+-----+-----+-----
plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
postgis | 2.2.1 | maps | PostGIS geometry, geography, and raster spatial types and functions
(2 rows)
=# ¥q
$ psql maps -U maps
=> ¥d
List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
public | geography_columns | view | postgres
public | geometry_columns | view | postgres
public | raster_columns | view | postgres
public | raster_overviews | view | postgres
public | spatial_ref_sys | table | postgres
(5 rows)

```

図 4.1.4.1: PostGIS 有効化

作成されたテーブルやビューは下表の通りです。

表 4.1.41: PostGIS を有効化することで作成されたテーブル、ビュー

No.	オブジェクトタイプ	オブジェクト名	概要
1	テーブル	spatial_ref_sys	空間参照系データを格納するテーブル。空間参照系間の変換や投影変換に使用されます。
2	ビュー	geography_columns	各テーブルの geography 属性とそのテーブルの SRID を表示します。
3	ビュー	geometry_columns	各テーブルの geometry 属性とそのテーブルの SRID を表示します。
4	ビュー	raster_columns	各テーブルの raster 属性とそのテーブルの SRID を表示します。
5	ビュー	raster_overviews	各テーブルの raster_overviews 情報を表示します。

4.2. データインポート

3章で shp2pgsql を用いて変換したデータを、作成した maps データベースにインポートを行います。

4.2.1. psql コマンドによるデータインポート

shp2pgsql にて変換したデータを PostgreSQL の psql コマンドを用いてインポートします。

```
$ ls -l *.dmp
-rw-rw-r-- 1 postgres postgres 583971 3月 15 21:59 2016 bunkasetu_kanagawa.dmp
-rw-rw-r-- 1 postgres postgres 327238 3月 15 21:59 2016 bunkasetu_tokyo.dmp
-rw-rw-r-- 1 postgres postgres 6092144 3月 15 20:15 2016 gyosei_kuiki_kanagawa.dmp
-rw-rw-r-- 1 postgres postgres 15057395 3月 15 20:15 2016 gyosei_kuiki_tokyo.dmp
-rw-rw-r-- 1 postgres postgres 780825 3月 15 16:05 2016 road_map_kanagawa.dmp
-rw-rw-r-- 1 postgres postgres 881165 3月 15 16:05 2016 road_map_tokyo.dmp
$ psql maps -f gyosei_kuiki_tokyo.dmp -U maps
SET
SET
BEGIN
CREATE TABLE
ALTER TABLE
          addgeometrycolumn
-----
maps.gyosei_kuiki.geom SRID:4612 TYPE:MULTIPOLYGON DIMS:2
(1 row)

COPY 6259
CREATE INDEX
COMMIT
ANALYZE
$
```

図 4.2.1.1: 東京都行政区域データ投入

下記に東京都の行政区域データをインポートした結果を確認します。

```
$ psql maps -U maps
=> \d
          List of relations
Schema |          Name          | Type   | Owner
-----+-----+-----+-----
maps   | geography_columns      | view   | postgres
maps   | geometry_columns       | view   | postgres
maps   | gyosei_kuiki           | table  | maps
maps   | gyosei_kuiki_gid_seq   | sequence | maps
maps   | raster_columns         | view   | postgres
maps   | raster_overviews       | view   | postgres
maps   | spatial_ref_sys        | table  | postgres
(7 rows)
```

図 4.2.1.2: 東京都の行政区域データ投入直後のデータベースの状態

東京都の行政区域のデータインポートにより、作成されたテーブルやビューは下表の通りです。

表 4.2.1.1: データ投入後に作成されたテーブル、ビュー

No.	オブジェクトタイプ	オブジェクト名	概要
1	テーブル	gyosei_kuiki	行政区域データテーブル。ユーザが任意で作成。テーブル名は shp2pgsql で変換の際に指定しています
2	シーケンス	gyosei_kuiki_gid_seq	gyosei_kuiki テーブルの gid カラムで serial 型で利用するシーケンス。

以上で東京都の行政区域のデータインポートは完了です。

上記と同様の手順で、他のデータに関してもインポートを実施して下さい。

本書では、下記コマンドで各データインポートを実施しました。

表 4.2.1.2: 変換済みデータインポートコマンド

No.	データ概要	都道府県	データインポートコマンド
1	行政区域データ	東京都	\$ psql maps -f gyosei_kuiki_tokyo.dmp -U maps
2		神奈川県	\$ psql maps -f gyosei_kuiki_kanagawa.dmp -U maps
3	道路データ	東京	\$ psql maps -f road_map_tokyo.dmp -U maps
4		神奈川県	\$ psql maps -f road_map_kanagawa.dmp -U maps
5	文化施設データ	東京都	\$ psql maps -f bunkasetu_tokyo.dmp -U maps
6		神奈川県	\$ psql maps -f bunkasetu_kanagawa.dmp -U maps

4.3. 地理情報データのインポート確認

PostgreSQL に投入されたデータは psql コマンドか PostGIS に対応した GIS 閲覧アプリケーションで表示が可能です。本章ではデスクトップ版のソフトウェアである「QGIS」を用いて、PostgreSQL データベースに接続して地図データを開覧します。

4.3.1. PostGIS レイヤ追加

QGIS から PostGIS の各テーブル内のデータを参照するため、PostGIS(PostgreSQL)をレイヤとして登録します。

① PostGIS レイヤ追加タブ押下

QGIS メニューから「レイヤ(L)」 「レイヤの追加」 「PostGIS レイヤの追加(Ctrl+Shift+D)」を選択します。

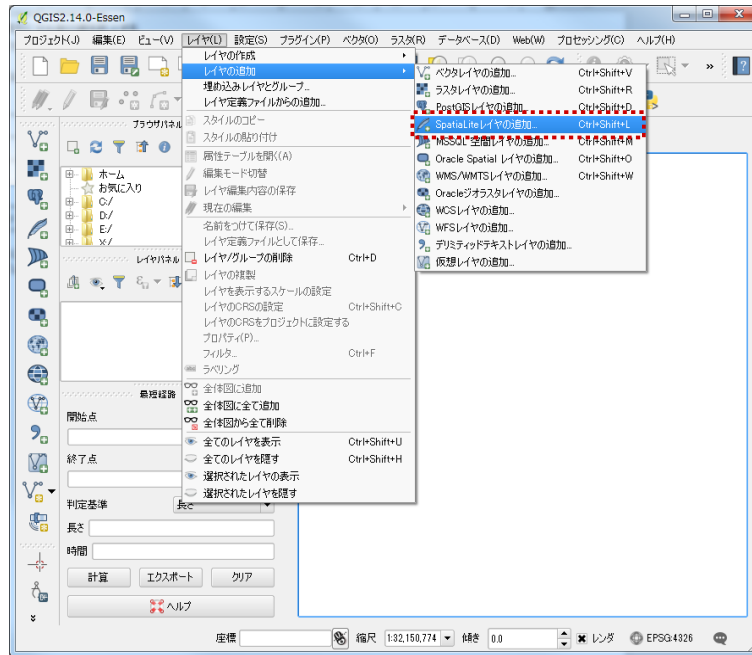


図 4.3.1.1: PostGIS レイヤ追加

② PostGIS レイヤ新規登録

「PostGIS テーブルを追加」ウィンドウが表示されます。左上の「新規」ボタンを押下して下さい。

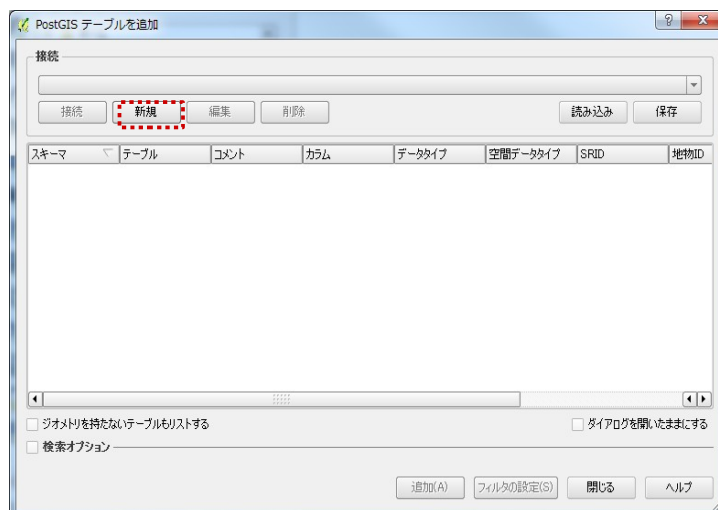


図 4.3.1.2: PostGIS 新規登録

③ PostGIS の接続情報登録

「PostGIS テーブルを追加」ウィンドウが表示されます。

名称、ホスト、データベース、ポート、SSL モード、ユーザ名、パスワードを設定します。

「テスト接続」をクリックして、接続できることを確認したら「OK」ボタンを押下して終了して下さい。



図 4.3.1.3: PostGIS の接続情報登録

④ PostGIS テーブル一覧表示

プルダウンリストから先ほど入力した名称(例:wg2)を選択し、「接続」をクリックします。

正しく設定されていた場合、テーブル一覧が表示されます。

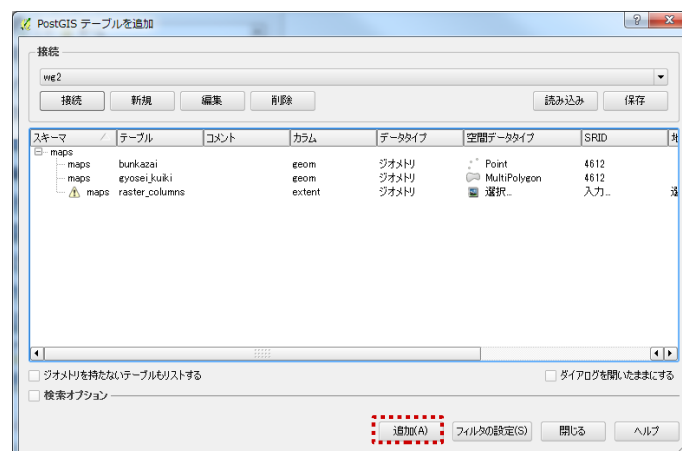


図 4.3.1.4: 作成済みテーブル一覧を表示

⑤ PostGIS テーブル追加

表示したいテーブルを選択し、「追加」ボタンを押下して下さい。

下記のようにデータベース内のテーブルが表示されます。

(下記例では、東京都および神奈川県を格納した gyosei_kuiki テーブルと、文化施設データを格納した bunkasetu テーブルを選択した状態「追加」ボタンを押下しています。)

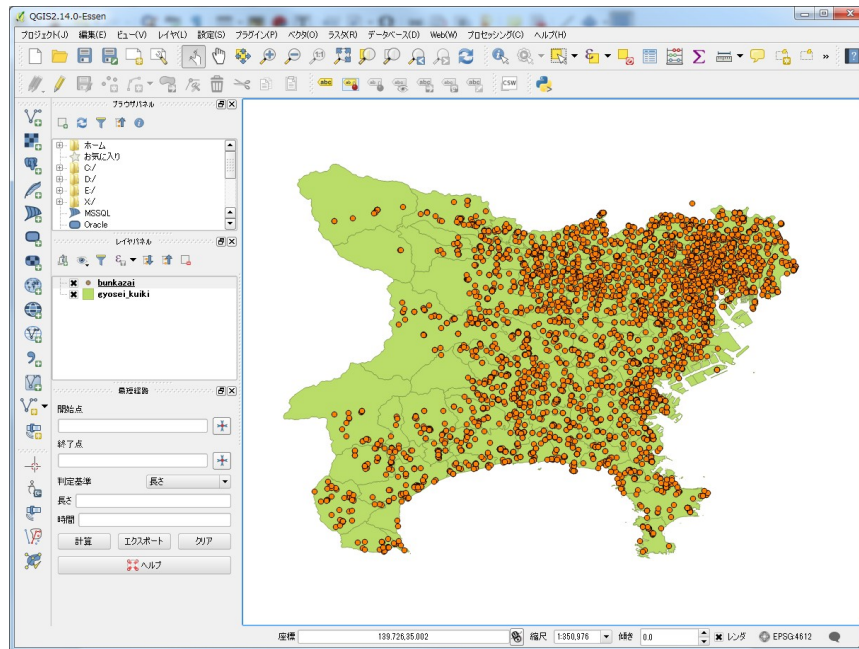


図 4.3.1.5: 東京都および神奈川県における文化施設の位置

以上で地理情報データのインポートは完了です。

5. 地理情報データ分析

本章では事前検討にて挙げた要件を満たすため、地理情報データを分析し可視化する手順を記載します。

5.1. 要件の確認

事前検討にて記載した要件は下表の通りです。下表の要件を満たすため、地理情報データを可視化します。

表 5.1.1: 要件の確認

No.	要件	概要
1	文化施設の多い行政区域の調査	地区内の文化施設を集計し、地図上に可視化します。
2	文化施設までの経路検索	ある地点から任意の文化施設までの経路を検索します。

5.2. 文化施設の多い行政区域の調査

本節では東京都と神奈川県各市町村毎に文化施設を集計し、地図上に可視化します。
 下表の手順で可視化を行いました。

表 5.2.1: 調査手順

No.	概要	概要	使用する関数
5.2.1	複数ポリゴンからなる行政区域を一つのポリゴンに整形	海や他の市町村によって分断されている行政区域は、複数のポリゴンからなるため、一つのポリゴンにまとめます。	ST_Union ST_Multi
5.2.2	行政区域毎の文化施設数をカウント	行政区域内に存在する文化施設をカウントします。	ST_Within
5.2.3	分析結果を QGIS 上で表示する	QGIS で分析結果を表示します。	—

使用する関数の詳細は下表の通りです。

表 5.2.2: 使用する関数

No.	関数名	概要
1	ST_Union	結合した総和を返す関数です。論理演算の「OR」に相当します。
2	ST_Multi	マルチポリゴンを返す関数です。 ポリゴンにはマルチポリゴンと単一ポリゴンがあり、別種類のポリゴンとして認識されるため、全てのポリゴンをマルチポリゴンに統一するために使用します。
3	ST_Within	ジオメトリが完全にジオメトリの内側にある場合に TRUE を返す関数です。

5.2.1. 複数ポリゴンからなる行政区域を一つのポリゴンに整形

下図の SQL を実行して、複数のポリゴンからなる行政区域を一つのマルチポリゴンにしたテーブルを作成します。

```
$ psql maps -U maps

maps=>
CREATE TABLE joined_gyosei_kuiki AS
SELECT citycode
      , gyosei_kuiki
      , ST_Union(geom) AS geom
FROM
  ( SELECT CONCAT(n03_001,' ', COALESCE(n03_003,' '), ' ', COALESCE(n03_004,' ')) AS gyosei_kuiki
    , n03_007 AS citycode
    , ST_Multi(geom) AS geom
  FROM gyosei_kuiki
  WHERE CONCAT(n03_004) <> ' 所属未定地'
  ) AS gk
GROUP BY gyosei_kuiki, citycode
ORDER BY citycode;
SELECT 120
```

図 5.2.1.1: 複数ポリゴンからなる行政区画を 1 つのマルチポリゴンとしたテーブル

作成したテーブル(joined_gyosei_kuiki)は下表の構成になります。

表 5.2.1.1: joined_gyosei_kuiki テーブル概要

No.	列名	概要	データ例
1	citycode	市町村コード	13101
2	gyosei_kuiki	市町村名 (行政区域名)	東京都 千代田区
3	geom	市町村のポリゴン データ	01060000200412000001000000010300000001000000700200008CC32251BB7 86140ACF95BE912DA414015159AB9BA78614099C8B7D7FFD9414085D5A65 1BB78614055A8CC9FFFD941400FB31BA6BB786140B9A6267FFFD94140D9E D2AD9BE786140C44F4150FED941402EE12B15BF7861407CDD3D3BFED941 40BC925E80BF78 (...)

5.2.2. 行政区域毎の文化施設数をカウント

行政区域毎に存在する文化施設の下図の SQL を実行します。

```
$ psql maps -U maps

maps=>
SELECT
  citycode,
  gyosei_kuiki,
  (
    (SELECT count(*) AS count
     FROM bunkasetu
     WHERE ST_Within(bunkasetu.geom, joined_gyosei_kuiki.geom)
    )::integer AS bunkasetu_cnt,
  geom
FROM joined_gyosei_kuiki;
※ 実行結果は省略
```

図 5.2.2.1: 行政区域毎の文化施設数をカウントする SQL

上記 SQL の実行結果は下表の通りです。(1 レコードのみ抜粋)

表 5.2.2.1: 行政区域毎の文化施設数をカウントする SQL 実行結果

No.	列名	概要	データ例
1	citycode	市町村コード	13101
2	gyosei_kuiki	市町村名 (行政区域名)	東京都千代田区
3	bunkasetu_cnt	行政区域内の文化 施設数	53
4	geom	地図空間情報	01060000200412000001000000010300000001000000700200008CC32251BB7 86140ACF95BE912DA414015159AB9BA78614099C8B7D7FFD9414085D5A65 1BB78614055A8CC9FFFD941400FB31BA6BB786140B9A6267FFFD94140D9E D2AD9BE786140C44F4150FED941402EE12B15BF7861407CDD3D3BFED941 40BC925E80BF78 (...)

5.2.3. 分析結果の可視化

本節では、これまでの各分析結果を「QGIS」を用いて、地図上に可視化する手順を記載します。
東京都と神奈川県の市町村毎に文化施設を集計した結果を QGIS を用いて、塗り分け地図(コロプレス図)を作成します。

① DB マネージャの起動

QGIS の「メニュー」から「データベース(D)」「DB マネージャ」「DB マネージャ」を選択します。

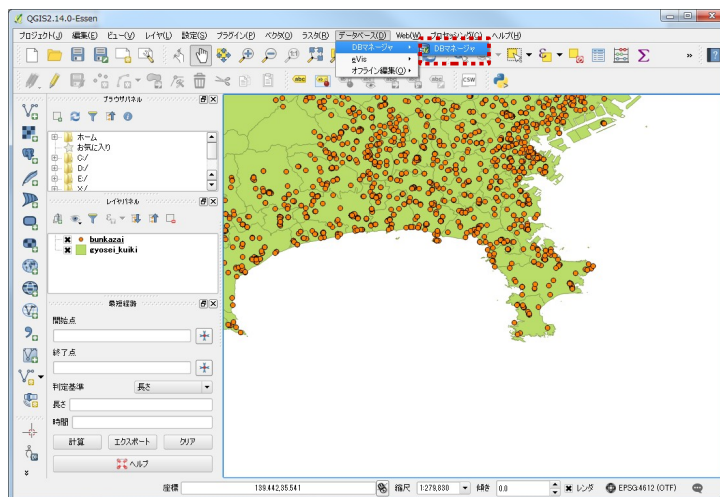


図 5.2.3.1: DB マネージャの起動

② SQL ウィンド の起動

ダイアログが表示されるのでリストから「PostGIS」を押下し「wg2」「maps」を選択して、「SQL ウィンド(S)」ボタンを押下します。

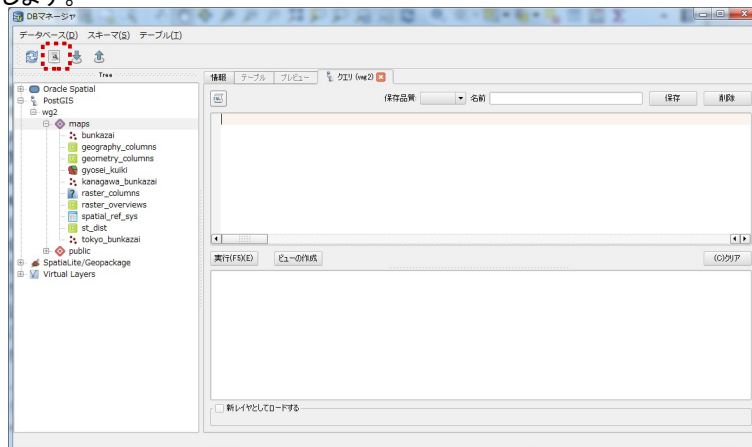


図 5.2.3.2:SQL ウィンドウを表示

③結果の反映

「5.2.2 行政区画毎の文化施設数をカウント」の SQL を貼り付け「実行(F5)(E)」ボタンを押下します。
SQL の実行結果が表示された後、「新レイヤとしてロードする」にチェックを入れ、「レイヤ名(接頭辞)」に任意の値(例:bunkasisetu_cnt)を入力し、「今ロードする！」ボタンを押下すると結果が反映されます。

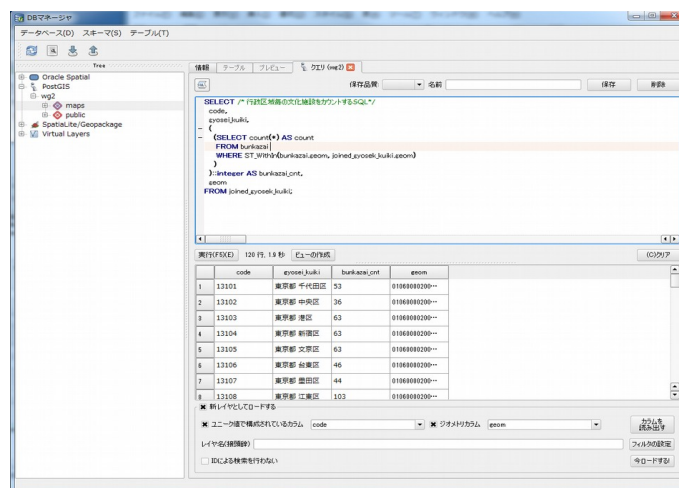


図 5.2.3.3SQL 実行結果を QGIS 上に反映

QGIS のメイン画面に戻るとロードしたレイヤ(bunkasisetu_cnt)が追加されています。
他のレイヤーにチェックが入っている場合は、チェックを外して下さい。

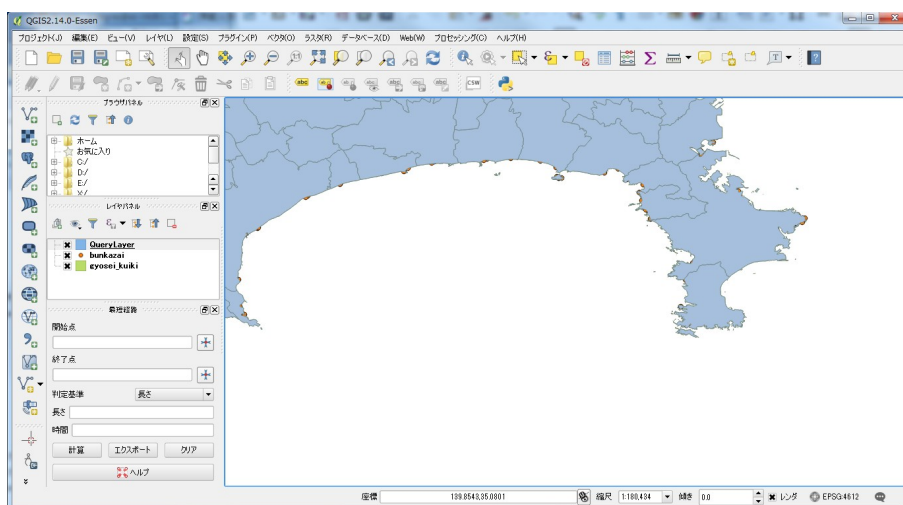


図 5.2.3.4:SQL 実行結果

④表示された地図データの加工

表示された地図データのプロパティを変更し、塗り分け地図(コロプレス図)を作成します。上記で作成したレイヤ(bunkasisetu_cnt)のプロパティを選択すると、ダイアログ画面が表示されます。

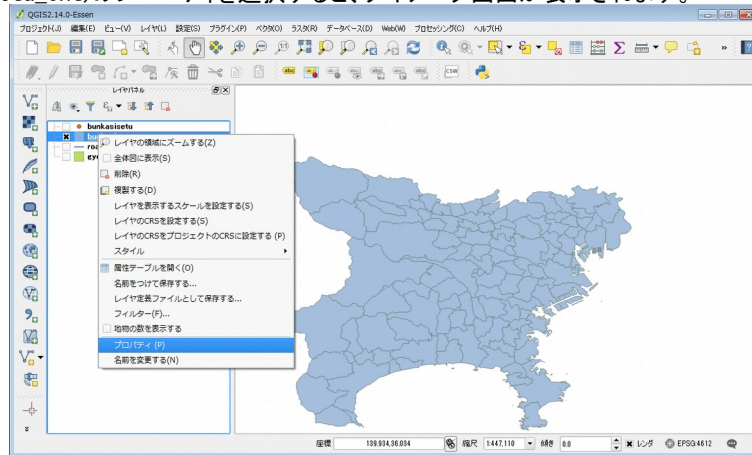


図 5.2.3.5:プロパティの選択

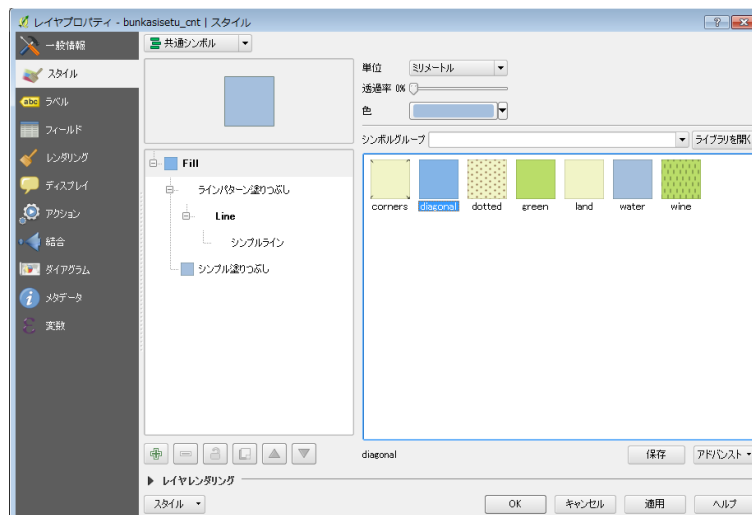


図 5.2.3.6:プロパティダイアログ

⑤値による色分け

「bunkasisetu_cnt」の値によってポリゴンを段階的に色分けします。ウィンドウの左側にある「スタイル」タブを押下し、「共通シンボル」とあるプルダウンリストから「段階に分けられた」を選択します。

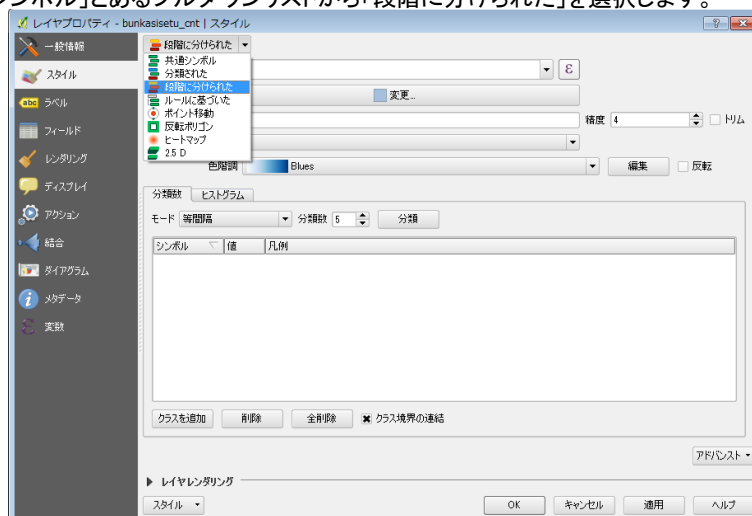


図 5.2.3.7: プロパティダイアログ。

⑥集計対象の Colum の選択

カラムプルダウンリストにて「bunkasetu_cnt」を選択し、「分類数」タブにある分類数に「7」を入力し、「OK」ボタンを押下すると塗り分け地図(コプロレス図)が表示されます。

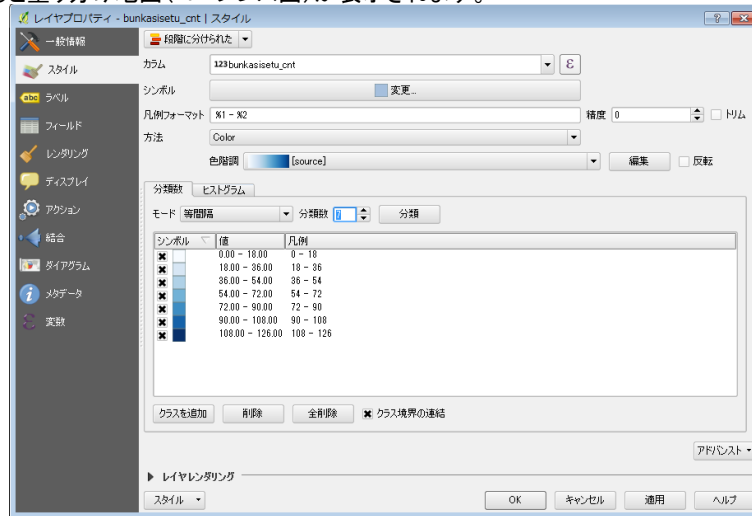


図 5.2.3.8:スタイルの選択



図 5.2.3.9 値と塗り分けの設定

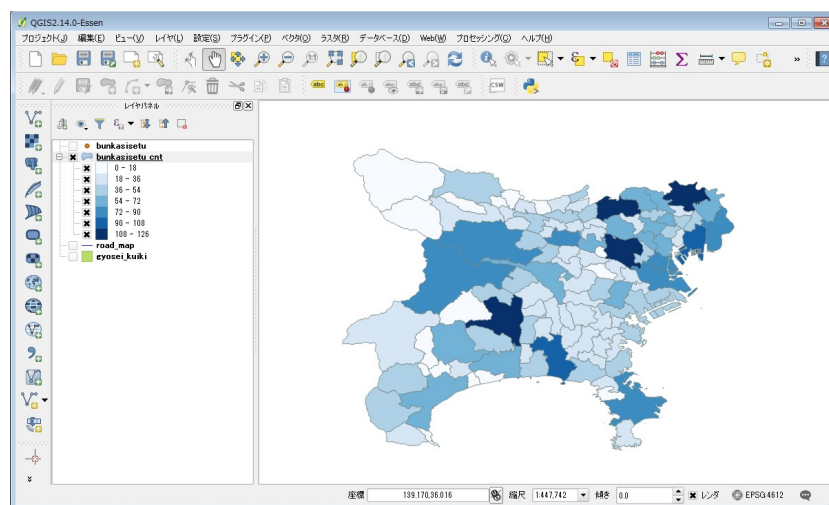


図 5.2.3.10:実行結果

⑦文化施設の多い地域を抽出する

塗り分け地図が表示されたので、より可読性を向上させるために文化施設の多い行政区域名を表示させます。QGISのデータベースマネージャーを開いて下図のSQLを入力し、レイヤ名を「gyosei_kuiki_name」とし、新レイヤーとしてロードしてください。

```
WITH cnt_table as (
SELECT citycode
, gyosei_kuiki
/* 行政区域内の文化施設を集計する */
, ((SELECT count(*) AS count
FROM bunkasetu
WHERE ST_Within(bunkasetu.geom, joined_gyosei_kuiki.geom)
)::integer AS bunkasetu_cnt
, geom
FROM joined_gyosei_kuiki)
/* WITH 句内で集計した文化施設数(bunkasetu_cnt)を WHERE 句で絞り込む */
SELECT citycode
, gyosei_kuiki
, bunkasetu_cnt
, geom
FROM cnt_table
/* 閾値は一番濃い青の範囲である 108 以上に設定 */
WHERE bunkasetu_cnt >= 108;
```

図5.2.3.11:文化施設の多い行政区域名を抽出するSQL

⑧行政区域名を表示させる

新規で作成された「gyosei_kuiki_name」レイヤを右クリックしてプロパティウィンドウを表示させます。プロパティウィンドウの左にあるラベルタブを選択しプルダウンリストから、「このレイヤのラベル表示」を選択してください。

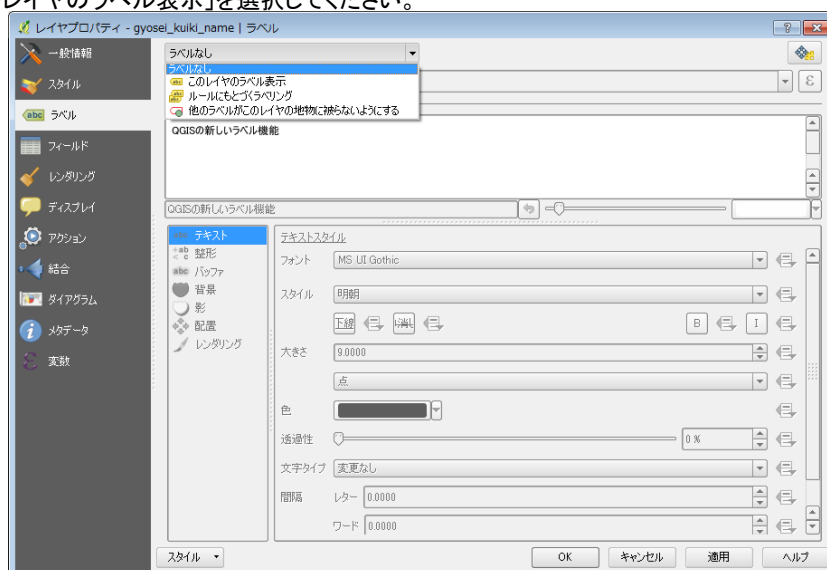


図 5.2.3.12: ラベルを表示する

ラベルにて「gyosei_kuiki」カラムを選択します。テキストや整形を好みに調整して、「適用」ボタンをクリックして下さい。文化施設が多い行政区域の名前だけが地図上に表示されます。

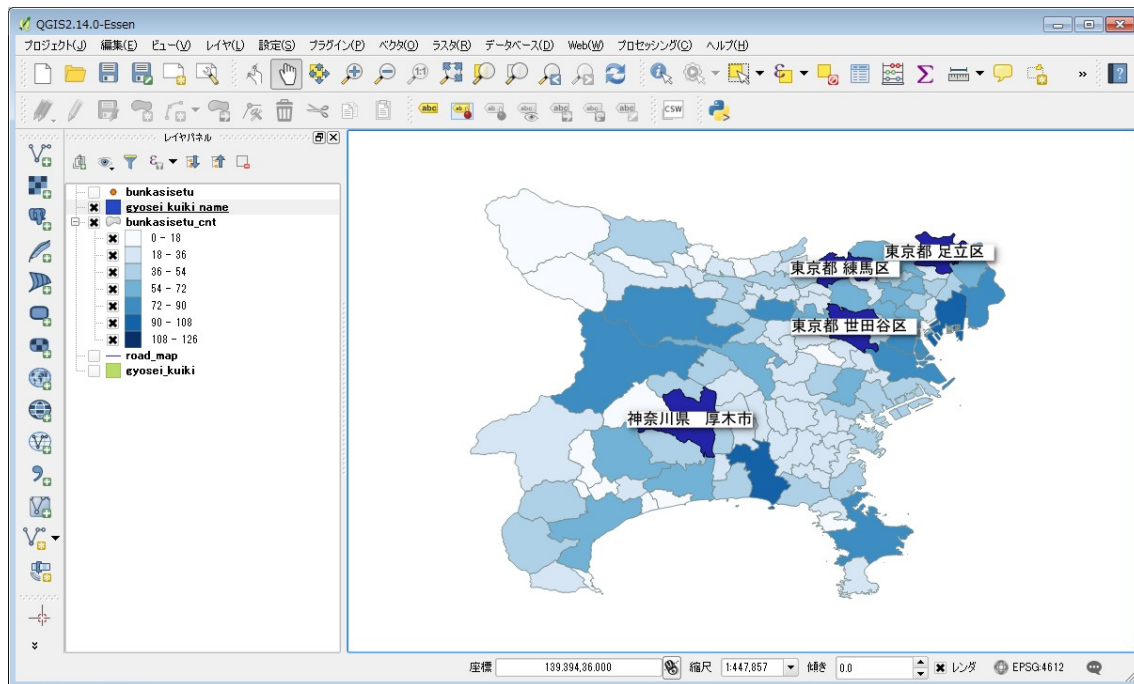


図:5.2.3.11:文化施設の多い行政区域名の出力

文化施設が多い行政区域は下表の通りです。

表 5.2.3.1:文化施設の多い行政区域一覧(降順)

No.	行政区域名	文化施設数
1	東京都足立区	126
2	神奈川県厚木市	121
3	東京都練馬区	114
4	東京都世田谷区	116

以上で、要件の1つである「文化施設の多い行政区域の調査」は完了です。
 この結果を受け、旅行会社Aは上表の行政区域の文化施設を巡る観光ルートを作成する事にしました。

5.3. 文化施設までの経路検索

本節では PostGIS に経路検索機能を追加する拡張モジュール「pgRouting」を用いて、文化施設間の最短経路図を作成します。作成手順は下表の通りです。

表 5.3.1:最短経路図の作成手順

記載節	手順	概要	使用する関数
5.3.1	pgRouting の有効化	pgRouting を有効化します。	–
5.3.2	経路情報の作成	ネットワークポロジ ² を作成します。	pgr_createTopology
5.3.3	道路の距離を計算	各道路の距離 ³ を算出します。	ST_Length
5.3.4	訪問する文化施設を決定	訪問する文化施設を決定し、そこから最も近い位置にある道路の頂点ノードを算出します。	–
5.3.5	訪問順序の算出	最も効率よく文化施設を訪問できる順序を算出します。	pgr_tsp
5.3.6	最短経路の検索	最短経路を検索します。	pgr_dijkstra
5.3.7	分析結果を QGIS 上で表示する	QGIS で分析結果を表示します。	–

使用する関数の詳細は下表の通りです。

表 5.3.2:使用する関数

No.	関数名	概要
1	pgr_createTopology	ネットワークポロジを作成する関数です。 今回利用する経路データには、どの経路がリンクしているかといった情報が存在しないため、本関数を使用します。
2	ST_Length	道路 (MULTILINESTRINGS ジオメトリ) の二次元長を返します。 最短経路を算出するにあたって使用するコスト情報として、本関数の戻り値を使用します。
3	pgr_tsp	巡回セールスマン問題 ⁴ を解く関数です。文化施設を巡る順序を求める際に使用します。
4	pgr_dijkstra	ダイクストラアルゴリズム ⁵ を使用し、2 点間の最短経路を返す関数です。

5.3.1. pgRouting の有効化

pgRouting は PostGIS とは別に「CREATE EXTENSION」コマンドを使用して有効化する必要があります。有効化するコマンドは下図の通りです。

```
$ psql maps -U postgres
=# CREATE EXTENSION pgrouting;
CREATE EXTENSION
=# \dx
```

List of installed extensions			
Name	Version	Schema	Description
pgrouting	2.0.0	public	pgRouting Extension
plpgsql	1.0	pg_catalog	PL/pgSQL procedural language
postgis	2.2.1	maps	PostGIS geometry, geography, and raster spatial types and functions

(3 rows)

図 5.3.1:pgRouting の有効化

2 ノード(頂点)とエッジ(道路)の接続関係を表現したものです。

3 本節では求めている距離は回転楕円体上の 2 点間距離であり、標高(楕円体高 + ジオイド高)などは考慮出来ておりません。

<http://www.gsi.go.jp/buturisokuchi/geoid.html>

4 巡回セールスマン問題 <https://ja.wikipedia.org/wiki/%E5%B7%A1%E5%9B%9E%E3%82%BB%E3%83%BC%E3%83%AB%E3%82%B9%E3%83%9E%E3%83%B3%E5%95%8F%E9%A1%8C>

5 ダイクストラ法 <https://ja.wikipedia.org/wiki/%E3%83%80%E3%82%A4%E3%82%AF%E3%82%B9%E3%83%88%E3%83%A9%E6%B3%95>

5.3.2. 経路情報の作成

国土情報ダウンロードサービスから取得した道路情報だけでは経路探索を行うことは出来ません。経路探索のためには、道路がどのように繋がっているかといった情報であるネットワークポロジ情報(以下、経路情報)を追加する必要があります。

下図のコマンドを実行して、経路情報を作成します。

```
$ psql maps -U maps
ALTER TABLE road_map ADD COLUMN "source" integer;
ALTER TABLE road_map ADD COLUMN "target" integer;
# 第一引数に道路テーブル名を指定
# 第二引数に頂点を同一とみなす許容範囲を指定
# 第三引数に道路形状を保持するカラム名を指定
# 第四引数に id となるカラム名を指定
=> SELECT pgr_createTopology('road_map', 0.0001, 'geom', 'gid');
# インデックスも自動で作成されます。
:
(省略)
:
pgr_createtopology
-----
OK
(1 row)

# ネットワークポロジの頂点を管理する別テーブルが作成されます。
=> \d road_map_vertices_pgr

          Table "maps.road_map_vertices_pgr"
   Column   |      Type      | Modifiers
-----+-----+-----
 id          | bigint         | not null default nextval('road_map_vertices_pgr_id_seq'::regclass)
 cnt         | integer        |
 chk         | integer        |
 ein         | integer        |
 eout        | integer        |
 the_geom    | geometry(Point,4612) |
Indexes:
    "road_map_vertices_pgr_pkey" PRIMARY KEY, btree (id)
    "road_map_vertices_pgr_the_geom_idx" gist (the_geom)
```

図 5.3.2.1: ネットワークポロジ情報の作成

上記処理により左図のような道路情報に、頂点と接続関係の情報が追加されます。(右図参照)

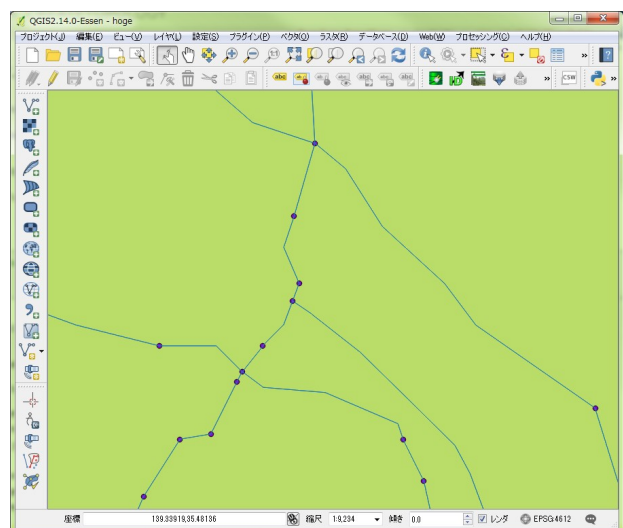
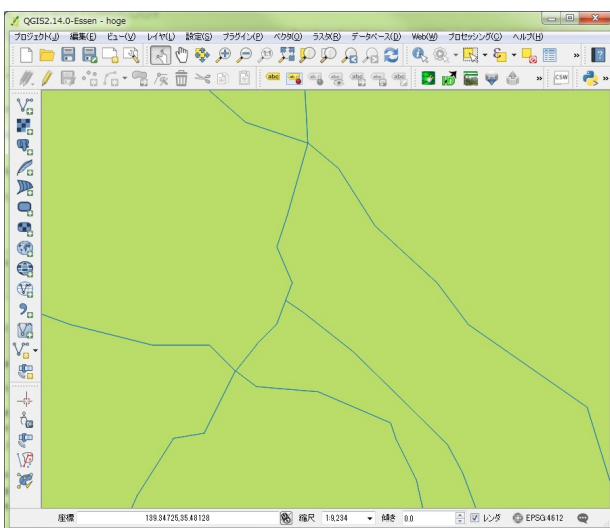


図 5.3.2.2: ネットワークポロジ情報の追加(図解)

5.3.3. 経路の距離を計算

最短経路を求める場合、経路を通る際のコスト(主に距離や時間)となる情報が必要となります。今回は経路の距離を使用します。

下図のコマンドを実行することにより経路の二次元長(メートル)を計算し、length カラムに格納します。

```
=> ALTER TABLE road_map ADD COLUMN "length" double precision;
=> UPDATE road_map SET length = ST_Length(geom::geography, true);
UPDATE 9628
```

図 5.3.3.1: 経路の距離を算出

ここまでの手順で経路情報テーブルは以下のようになります。

表 5.3.3.1: road_map テーブル概要

No.	列名	型	格納データ
1	gid	serial	レコードを一意に特定するための ID
2	n01_001	varchar (10)	道路種別コード
3	n01_002	varchar (20)	路線名
4	n01_003	varchar (20)	線名
5	n01_004	varchar (20)	通称
6	geom	geometry (MultiLineString, 4612)	道路の形状
7	source	integer	道路の始点となる頂点の ID (5.3.2 で作成されるテーブルの id)
8	target	integer	道路の終点となる頂点の ID (5.3.2 で作成されるテーブルの id)
9	length	double precision	コストとなる距離情報(メートル単位)

5.3.4. 訪問する文化施設を決定

訪問する文化施設を「bunkasetu」テーブルの中から選択します。ここでは、前節で求めた行政区域から、石洞美術館（東京都足立区）、唐澤博物館（東京都練馬区）、齋田記念館（東京都世田谷区）、森の民話館（神奈川県厚木市）とします。

また、2 点間の距離を求める演算子「<->」を使用し、各文化施設から一番近い経路の頂点を求めます。

各地点に相当する座標 (geography) を求めます。

```
# 石洞美術館の座標を検索
=> SELECT * FROM bunkasetu WHERE p27_005 = '石洞美術館';
gid | p27_001 | p27_002 | p27_003 | p27_004 | p27_005 | p27_006 | p27_007 | p27_008 | p27_009 |
geom
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
609 | 13121   | 3       | 99999   | 03001   | 石洞美術館 | 千住橋戸町 23 | 4 | -99 | 2006 |
0101000020041200006FA0C03B79796140919A7631CDDE4140
(1 row)

# 唐澤博物館の座標を検索
=> SELECT * FROM bunkasetu WHERE p27_005 = '唐澤博物館';
gid | p27_001 | p27_002 | p27_003 | p27_004 | p27_005 | p27_006 | p27_007 | p27_008 | p27_009 |
geom
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
180 | 13120   | 3       | 03002   | 03002   | 唐澤博物館 | 豊玉北 3-5-5 | 4 | 3 | 1993 |
010100002004120000CEG64ACC33756140E10A28D4D3DD4140
(1 row)

# 齋田記念館の座標を検索
=> SELECT * FROM bunkasetu WHERE p27_005 = '齋田記念館';
gid | p27_001 | p27_002 | p27_003 | p27_004 | p27_005 | p27_006 | p27_007 | p27_008 | p27_009 |
geom
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
158 | 13112   | 3       | 03002   | 03002   | 齋田記念館 | 代田 3-23-35 | 4 | -99 | 1997 |
010100002004120000BB6070CD1D7561407D0569C6A2D34140
(1 row)

# 森の民話館の座標を検索
=> SELECT * FROM bunkasetu WHERE p27_005 = '森の民話館';
gid | p27_001 | p27_002 | p27_003 | p27_004 | p27_005 | p27_006 | p27_007 | p27_008 | p27_009 |
geom
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2687 | 14212   | 3       | 03002   | 03002   | 森の民話館 | 七沢 901 (七沢森林公園) | 2 | 1 | 1989 |
0101000020041200003EB2B96A9E6961406891ED
7C3FB94140
(1 row)
```

図 5.3.4.1: 訪問する文化施設の検索

求めた geography をもとに、以下コマンドで経路情報の最短頂点（一番近い道路の頂点座標）の ID と距離（総距離の計算に使用）を求めます。

```
# 石洞美術館から一番近い頂点ノードを検索
=> SELECT id, the_geom, the_geom <-> '0101000020041200006FA0C03B79796140919A7631CDDE4140'::geography AS
distance
FROM road_map_vertices_pgr
ORDER BY distance LIMIT 1;
```

id	the_geom	distance
3201	010100002004120000CF23F386837961404010625F99DE4140	209.247523755815

(1 row)

```
# 唐澤博物館から一番近い頂点ノードを検索
=> SELECT id, the_geom, the_geom <-> '010100002004120000CE64ACC33756140E10A28D4D3DD4140'::geography AS
distance
FROM road_map_vertices_pgr
ORDER BY distance LIMIT 1;
```

id	the_geom	distance
2477	0101000020041200009A03CAC72A756140D4D0276EECD4140	129.774872723025

(1 row)

```
# 齋田記念館から一番近い頂点ノードを検索
=> SELECT id, the_geom, the_geom <-> '010100002004120000BB6070CD1D7561407D0569C6A2D34140'::geography AS
distance
FROM road_map_vertices_pgr
ORDER BY distance LIMIT 1;
```

id	the_geom	distance
2440	010100002004120000835F13671F756140D4F523C99CD34140	26.9173974565555

(1 row)

```
# 森の民話館から一番近い頂点ノードを検索
=> SELECT id, the_geom, the_geom <-> '0101000020041200003EB2B96A9E6961406891ED7C3FB94140'::geography AS
distance
FROM road_map_vertices_pgr
ORDER BY distance LIMIT 1;
```

id	the_geom	distance
5260	01010000200412000093AED3277D696140308C8907F3B84140	450.104493575256

(1 row)

図 5.3.4.2: 最短頂点の検索

上記結果をまとめると以下のようになります。

表 5.3.4.1: 訪問地点の最短頂点の id と距離

No.	訪問地点	最短頂点の id	距離
1	石洞美術館	3201	209.247523755815
2	唐澤博物館	2477	129.774872723025
3	齋田記念館	2440	26.9173974565555
4	森の民話館	5260	450.104493575256

5.3.5. 訪問順序の算出

前項で求めた訪問地点について、どの順序で訪問すべきかを pgr_tsp 関数⁶を用いて算出します。今回は海外旅行者を想定しているため、成田空港から一番近い石洞美術館を開始点/終了点とします。

```
# 第一引数の SQL で訪問対象である道路の頂点のリストを検索
# 第二引数で開始地点を指定（第三引数を省略した場合は開始地点に戻るものとみなされる）
=> SELECT *
FROM pgr_tsp('SELECT id::integer, ST_X(the_geom)::float8 AS x, ST_Y(the_geom)::float8 AS y
FROM road_map_vertices_pgr
WHERE id = 3201
OR id = 2477
OR id = 2440
OR id = 5260
ORDER BY id',
3201);
```

seq	id1	id2	cost
0	2	3201	0.161855808242191
1	0	2440	0.419003389123057
2	3	5260	0.465430107704
3	1	2477	0.135935834926944

(4 rows)

図 5.3.5.1: 訪問順序の算出

上記結果から、「石洞美術館」「齋田記念館」「森の民話館」「唐澤博物館」の順に訪問することに決定しました。

⁶ pgr_tsp マニュアル <http://docs.pgrouting.org/2.0/en/src/tsp/doc/index.html>

5.3.6. 最短経路の検索

前項で求めた順序で訪問する際の最短経路を、ダイクストラアルゴリズムを用いて算出します。用いる関数は `pgr_dijkstra` 関数⁷になります。

```
=> SELECT id1 AS node, id2 AS edge, cost, b.geom
FROM
  (SELECT id1, id2, cost FROM pgr_dijkstra('
    SELECT gid AS id,
      source::integer,
      target::integer,
      length::double precision AS cost
    FROM road_map',
      3201, 2440, false, false)
  UNION ALL
  SELECT id1, id2, cost FROM pgr_dijkstra('
    SELECT gid AS id,
      source::integer,
      target::integer,
      length::double precision AS cost
    FROM road_map',
      2440, 5260, false, false)
  UNION ALL
  SELECT id1, id2, cost FROM pgr_dijkstra('
    SELECT gid AS id,
      source::integer,
      target::integer,
      length::double precision AS cost
    FROM road_map',
      5260, 2477, false, false)
  UNION ALL
  SELECT id1, id2, cost FROM pgr_dijkstra('
    SELECT gid AS id,
      source::integer,
      target::integer,
      length::double precision AS cost
    FROM road_map',
      2477, 3201, false, false)
  ) a LEFT JOIN road_map b ON (a.id2 = b.gid);
 node | edge |      cost      |
-----+-----+-----+
 3201 | 3363 | 287.446571361841 |
01050000200412000001000000010200000002000000CF23F386837961404010625F99DE414052224FB37C79614088F1DC7347DE4
140
 3200 | 3362 | 159.065922489441 |
0105000020041200000100000001020000000200000052224FB37C79614088F1DC7347DE4140BA7E2499737961409BE91C0B23DE4
140
 3180 | 3337 | 553.838467531458 |
01050000200412000001000000010200000003000000BA7E2499737961409BE91C0B23DE41400C37431E5F79614076427D05C8DD4
140F493A44A5879614
01CB2A2829ADD4140
(以下略)
```

図 5.3.6.1: 最短経路の算出

⁷ `pgr_dijkstra` マニュアル <http://docs.pgrouting.org/2.0/en/src/dijkstra/doc/index.html#pgr-dijkstra>

集約関数を使用し経路の距離の合計を求めます。

```
=> SELECT sum(cost) AS distance
FROM
(
  (上図と同様のため省略)
) a LEFT JOIN road_map b ON (a.id2 = b.gid);
      distance
-----
145370.977479584
(1 row)
```

図 5.3.6.2: 経路の距離の算出

「5.3.4 訪問する文化施設を決定」で算出した文化施設から道路までの距離も加算し、総距離を算出します。

```
総距離 = 209.247523755815
        + 129.774872723025
        + 26.9173974565555
        + 450.104493575256
        + 145370.977479584 = 146187.021767 (約 146km)
時速 50km/h の場合の概算時間 (単位 : 分) 、 = 146187.021767 / 1000 / 50 * 60 = 175.42442612 (3 時間弱)
```

図 5.3.6.3: 総距離の算出

5.3.7. 分析結果の可視化

本節では前節までに算出した最短経路を、QGISを用いて地図上に可視化します。

①文化施設の表示

初期画面として、「road_map」「bunkasisetu」「gyosei_kuiki」の順でテーブルをレイヤとして追加し表示します。

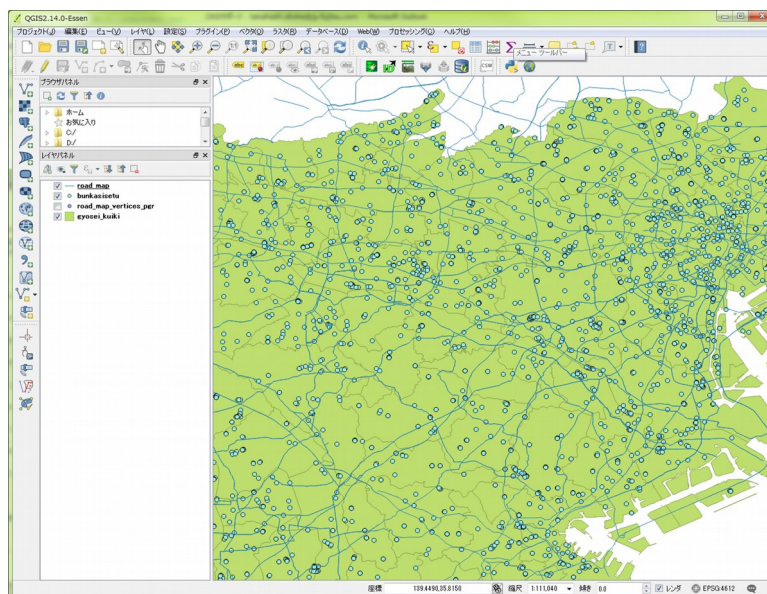


図 5.3.7.1: 文化施設の表示

②表示する文化施設の絞込

余分な文化施設が表示されないように絞り込んでいきます。

まずはレイヤパネルから「bunkasisetu」を右クリックし、「プロパティ(P)」を選択します。

設定画面が立ち上がるため、左メニューから「ラベル」を選択し、画面上部の「ラベル」の項目に文化施設の名称が格納されているカラムである「p27_005」を設定します。

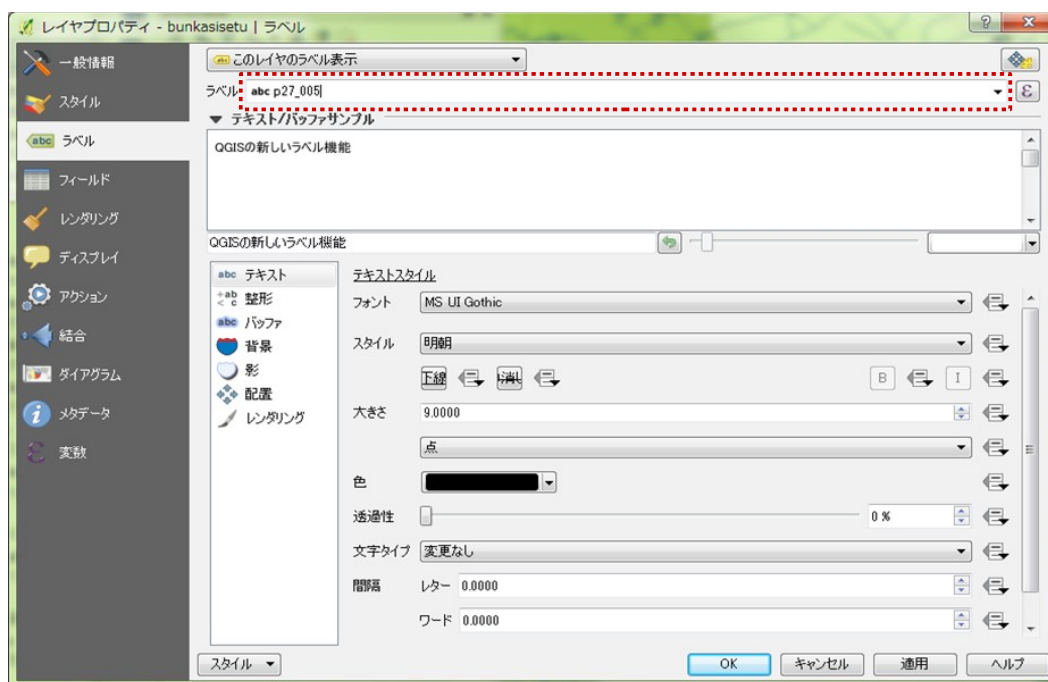


図 5.3.7.2: ラベル表示項目設定

次に左メニューの「スタイル」を選択し、設定を「ルールに基づいた」に変更します。

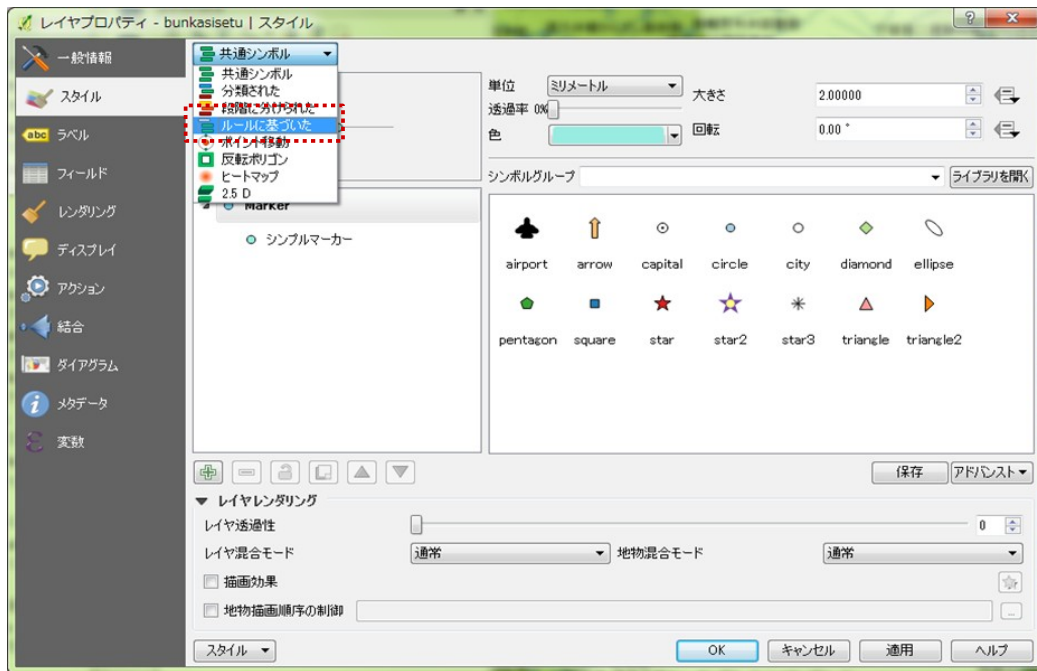


図 5.3.7.3: スタイル設定

「+」ボタンを押下し、文化施設を表示するルールを追加していきます。

ラベルにルールの名称、フィルターに検索条件をそれぞれ記入し、「OK」を押下します。

文化施設は gid で一意に定まるため、gid で条件を記載します。各文化施設の gid については「5.3.4 訪問する文化施設を決定」の図を参照して下さい。

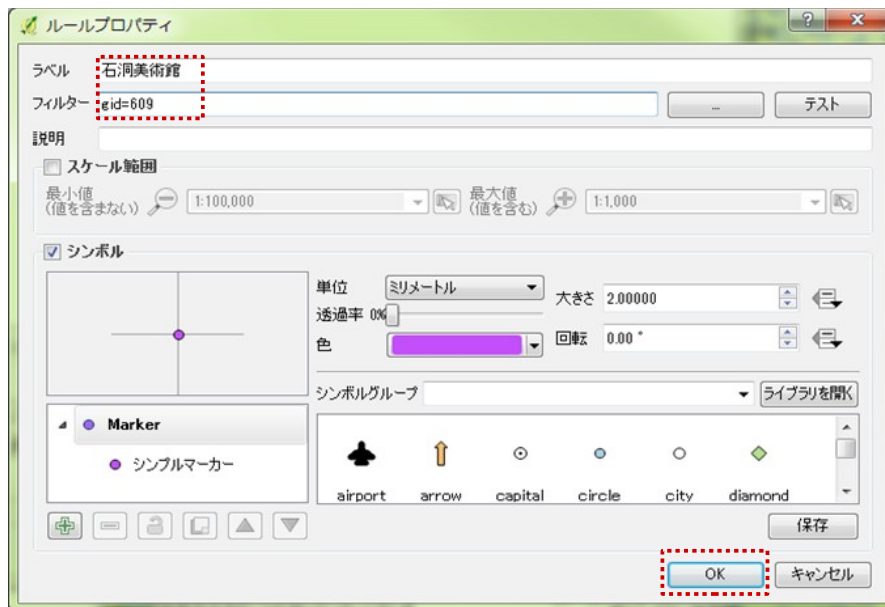


図 5.3.7.4: ルールの追加

上記ルールを全ての訪問地点について定義すると、以下のようになります。

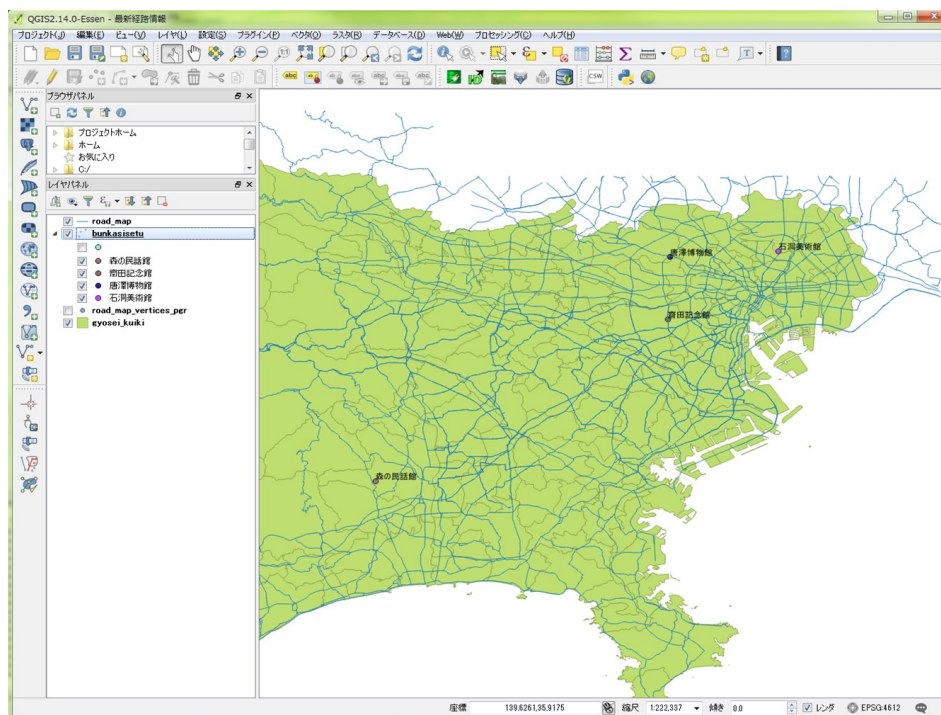


図 5.3.7.5: 訪問地点のみ表示

③最短経路の描画

DB マネージャから「5.3.6 最短経路の検索」にある経路を求めるクエリの結果をレイヤとして追加します。DB マネージャの操作に関しては「5.2.3 分析結果の可視化」の①～③を参照してください。

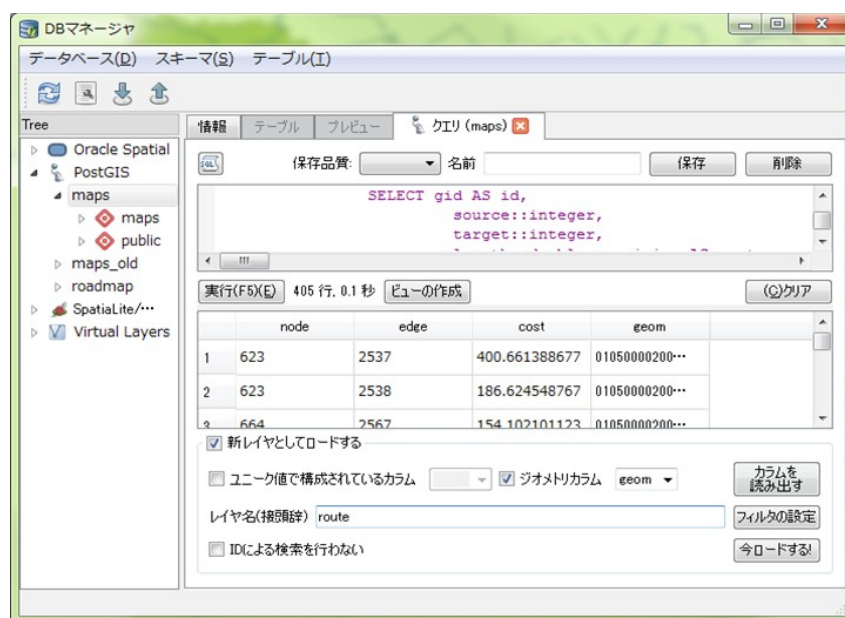


図 5.3.7.6: DB マネージャによるレイヤ追加

最短経路を赤線で描画した結果が以下になります。
 描画したことにより、「齋田記念館」が往復する経路上に存在することが判明しました。そのため「森の民話館」と訪問順序を入れ替えても移動時間に変化はないと思われます。

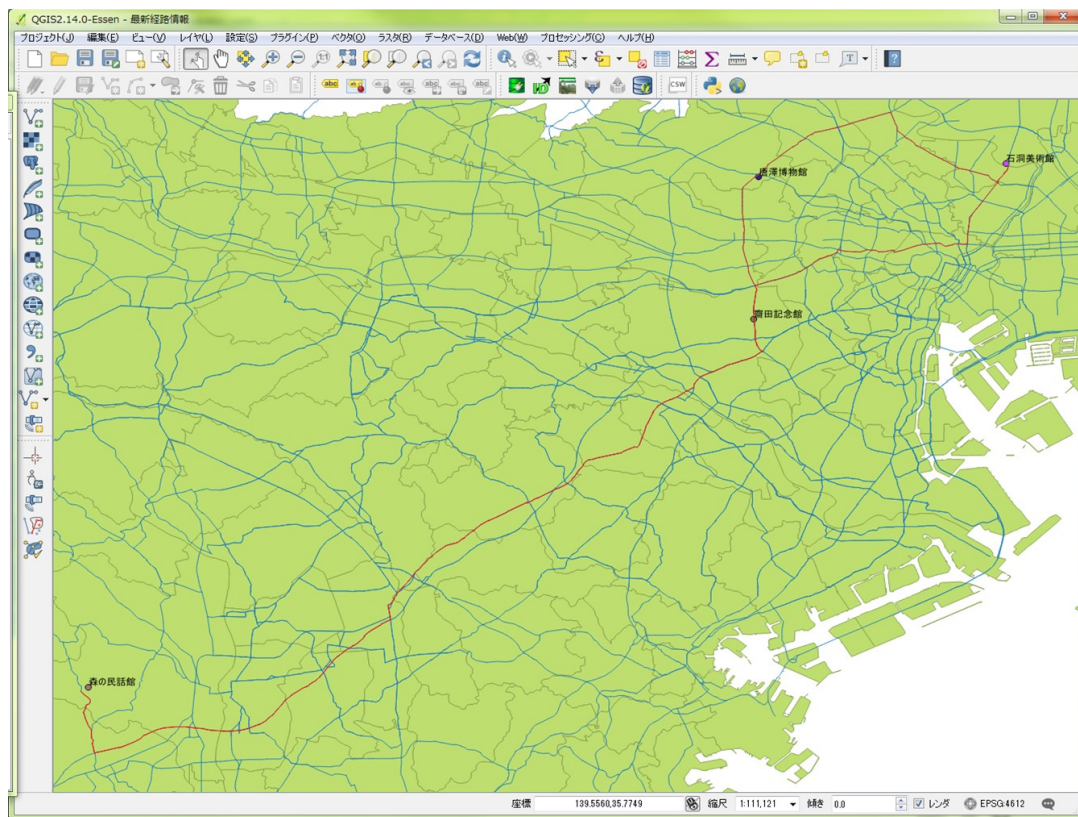


図 5.3.7.7: 最短経路の描画

以上で、要件の2つ目である「文化施設までの経路検索」はひとまず完了です。旅行会社 A では、これらの結果を踏まえて旅行プランを検討することになります。

所感

GISに求められる要件を例示し、PostGISを用いて要件を実現するまでの手順について記載しました。本書執筆に当たり、GISを実現するには地理情報に関する学術的な深い理解が必要と考えておりましたが、PostGISを利用することで有る程度の理解で、PostGISを用いてGISを実現することが出来ると、感じました。とりわけ、PostgreSQLに明るい技術者は、文字列や数値同様に地理情報がSQLで扱えることで、GISの理解が深まりやすいと感じます。

また、PostGISは日本語化されたマニュアルが公開されていたり、国内外の利用や導入に関する知見が、インターネット上に散見されるので、実現したい要件が定まっていれば、目的に沿った機能を簡単に探すことが可能でした。

本書を参照をいただくことで、PostGISを用いたGIS要件へのアプローチ方法をイメージできると考えておりますが、本資料は例示した要件の実現やPostGISの基本的な機能操作に主眼を置いているため、下表に示すような実際の運用を踏まえたうえでの記載やフォローが補足しています。本書を参考にPostGISを用いて地理情報データのデータ分析を行う場合は留意して下さい。

表 本資料に記載していない課題

No.	課題名	概要・留意点
1	地理情報データのライセンス (オープンデータのライセンス)	既存の地理情報データを利用する際は、提供元が定めたライセンスに従う必要があります。

著者

版	所属企業・団体名	部署名	氏名
DB選定基準編 第1.0版 (2015年度 WG2)	株式会社富士通ソーシャルサイエ ンスラボラトリ	公共ビジネス本部第三システム部	小山田 政紀
			高橋 勝平
			香田 紗希
			青木 俊彦