名古屋大学 COI 高精度地図フォーマット地図の使い方

2017 年 9 月 6 日版

文書の概要

この文書は，名古屋大学 COI 高精度地図フォーマットの地図情報を RDBMS に格納し，検索に利用するための手順を解説したものです．本来であれば，地図情報と動的情報は，ともにダイナミックマッププロトタイプシステムを介して統合的に扱われるものですが, 名古屋大学 COI 高精度地図フォーマットになれていただくために，地図情報単体でも使えるようにしております.

対象環境

Ubuntu Linux （バージョン 14.04-server-amd64 で確認)
(Windows 版の Postgres + PostGIS の組合せでも使えます．Mac 版では確認しておりません)

PostgreSQL と PostGIS のインストール

ここでは，名古屋大学 COI 高精度地図フォーマットの地図情報を格納する先として，PostgreSQL を使用します．PostgreSQL はオープンソースの RDBMS です．今回は PostgreSQL に空間検索の機能を入れるための拡張である PostGIS を追加でインストールします.

sudo を使ってルート権限を行使できるユーザで，以下のコマンドを実行してください.

1. PostgreSQL のインストール(apt-get コマンド使用)
   % sudo apt-get install –y postgresql postgresql-contrib libpq-dev

2. PostGIS のインストール(apt-get コマンド使用)
   % sudo apt-get install –y postgis postgresql-9.3-postgis-2.1 liblwgeom-dev libproj-dev

途中で apt-get から何か聞かれた場合は y と回答してください.

PostgreSQL の設定

データベース全体の管理者アカウントとして，通常はユーザ名 postgres がインストール時に作成されます．この管理者アカウントとは別に通常のデータベース操作を行うための一般アカウントを作成します.

1. su コマンドでユーザを postgres へ変更します(sudo 権限が必要)
   % sudo su – postgres

2. 新規アカウントを作成します（ここでは mirai を新規アカウント名としています)

% createuser –d –r –P　mirai
（アカウントのパスワードを設定するように言われるので 2 回入力してください）

3. 新規アカウント専用のデータベースを作成します(ここでは mirai_db をデータベース名としています)
% createdb –O mirai　mirai_db

4. 新規アカウント専用データベース（ここでは mirai_db）に，PostGIS の機能をロードします．以下の 2 つのコマンドをそれぞれ実行します．
% psql –c “create extension postgis; ”　mirai_db
% psql –c “create extension postgis_topology;”　mirai_db

5. スキーマサーチパスの設定を行います．以下のコマンドを実行します
% psql –c "alter user mirai set search_path=coi,public,topology;"　mirai_db

6. postgres から su 前のユーザに戻ります．
% exit

ネットワーク経由でのデータベース接続の許可

インストール直後の設定では，PostgreSQL をインストールしたマシンからのローカルな接続しか許可されません．ネットワーク経由で PostgreSQL にアクセスする場合は，接続許可のための設定を行う必要があります．ローカルからの接続しか行わない場合は，ここの設定手順は実施不要です．

設定ファイルは /etc/postgresql/9.3/main/ にあります（バージョンによっては途中の数字が異なります）．

1. sudo を使って設定ファイル pg_hba.conf を編集用に開きます．
% sudo vi /etc/postgresql/9.3/main/pg_hba.conf

2. pg_hba.conf の末尾に追加します．以下の 1 行は，プライベートネット 192.168.12.0/24 内のマシンからの全ユーザの接続をパスワード付で許可する設定です．
host　　all　　all　　192.168.12.0/24　　md5

3. sudo を使って設定ファイル postgresql.conf を編集用に開きます．
% sudo vi /etc/postgresql/9.3/main/postgresql.conf

4. postgresql.conf の中の，listen_addresses の設定を PostgreSQL サーバの IP アドレスに置き換えます．
（修正前）#listen_addresses = ‘localhost’
　　↓
（修正後 1）listen_addresses = ’192.168.12.4’　(固定 IP アドレスのときは IP アドレスを書きます)
（修正後 2）listen_addresses = ’*’　　(動的 IP アドレス(DHCP)のときは，*を書きます)

5. PostgreSQL を再起動します．
% sudo service postgresql restart

地図データの挿入

名古屋大学 COI 高精度地図フォーマットの道路地図データを PostgreSQL に挿入するための手順を説明します.

名古屋大学 COI 高精度地図フォーマットのインポート

PostgreSQL にはテーブルを外部に出力する(dump する)機能があります．その機能を使って他の環境の PostgreSQL から名古屋大学 COI 高精度地図フォーマットの dump データをもらってくると，ここに記述した方法でデータをインポートすることができます.

インポート手順

1. PostgreSQL の dump データを取得してください.

2. Zip ファイルを展開してください．(テキストファイルができます)
   % unzip pg_dump-20170731.zip

3. データを挿入するためのコマンドを実行します
   % psql –U mirai –f ./pg_dump-20170731.txt　mirai_db
   (ユーザ mirai で，専用データベース mirai_db にデータを取り込む場合)

作成されるテーブルについて

名古屋大学 COI 高精度地図フォーマットの各テーブルは 「coi.(テーブル名)」として作成されます．テーブル一覧を確認する場合は，以下のコマンドを使用してください.

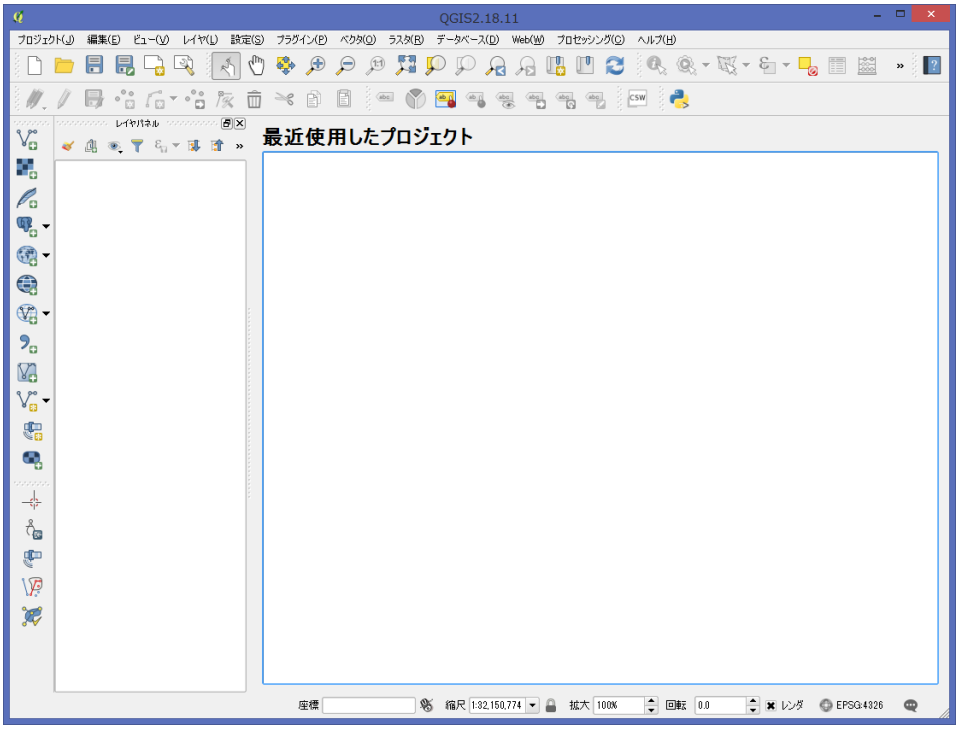
% psql –U mirai –c '¥dt *.*'　mirai_db
(ユーザ mirai で，専用データベース mirai_db の中のテーブルを全表示します)

地図データの表示

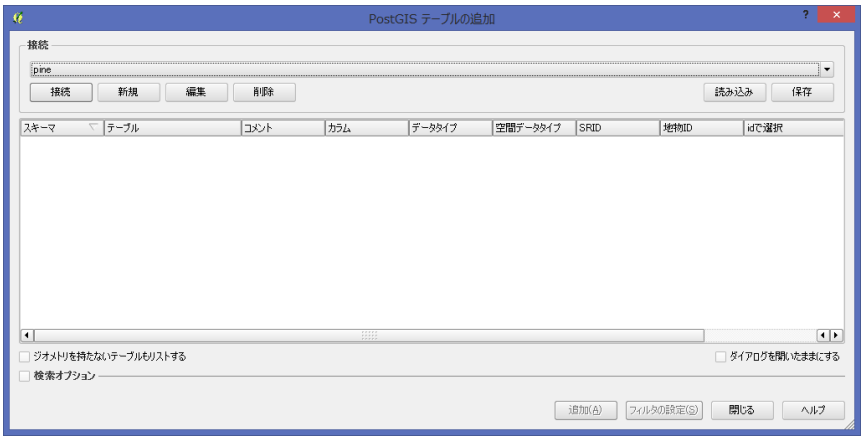
PostgreSQL に格納された空間データを表示する場合，PostGIS のデータ型に対応した GIS (Geographical Information System)ツールを使うと便利です．ここでは，フリーの GIS ソフト QGIS (http://qgis.org/ja/site/) を使用する場合を紹介します．

QGIS のインストール方法は，QGIS のマニュアルを参照してください．

1. QGIS を起動します．
   メニューから「レイヤ(L)」を選び，「レイヤの追加」から「PostGIS レイヤの追加」を選びます．

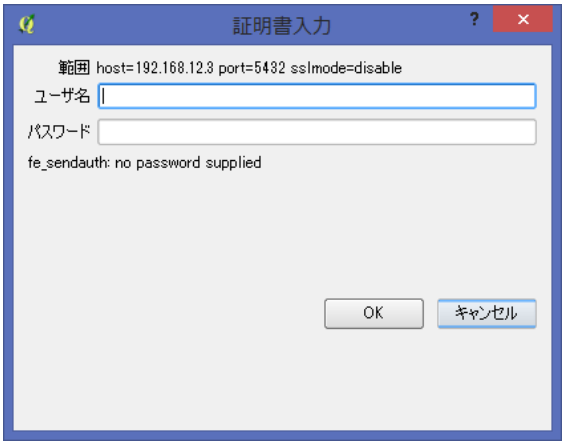


2. 下記のウインドウが出ますので，「新規」ボタンを押して，新規 PostGIS 接続の作成画面を出します．

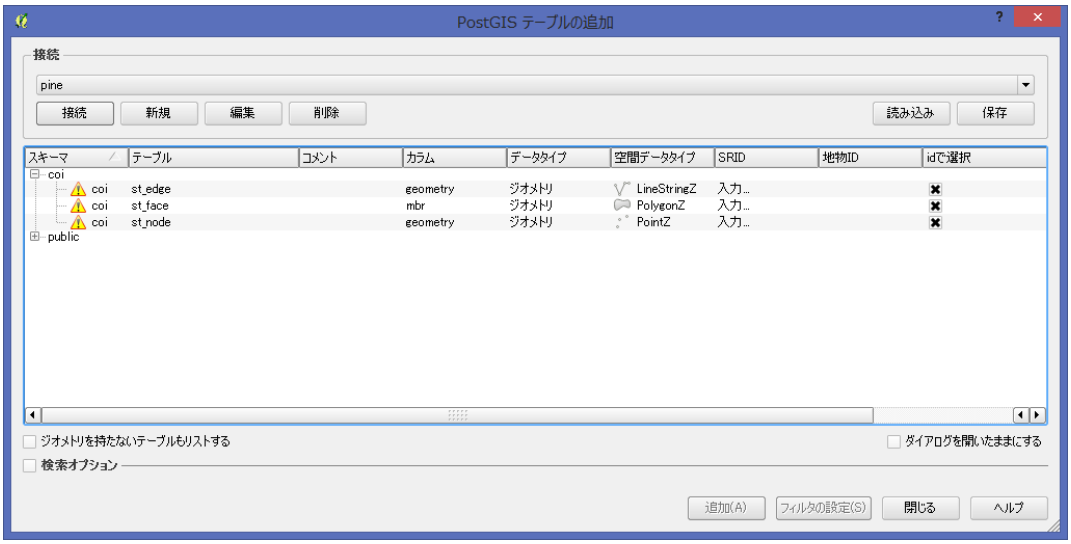


3. 「名称」(この接続につける名前）と「ホスト」(または IP アドレス)と「データベース」(データベース名)を入力します．「OK」ボタンを押すと 2.の画面に戻りますので，今度は接続ボタンを押します．

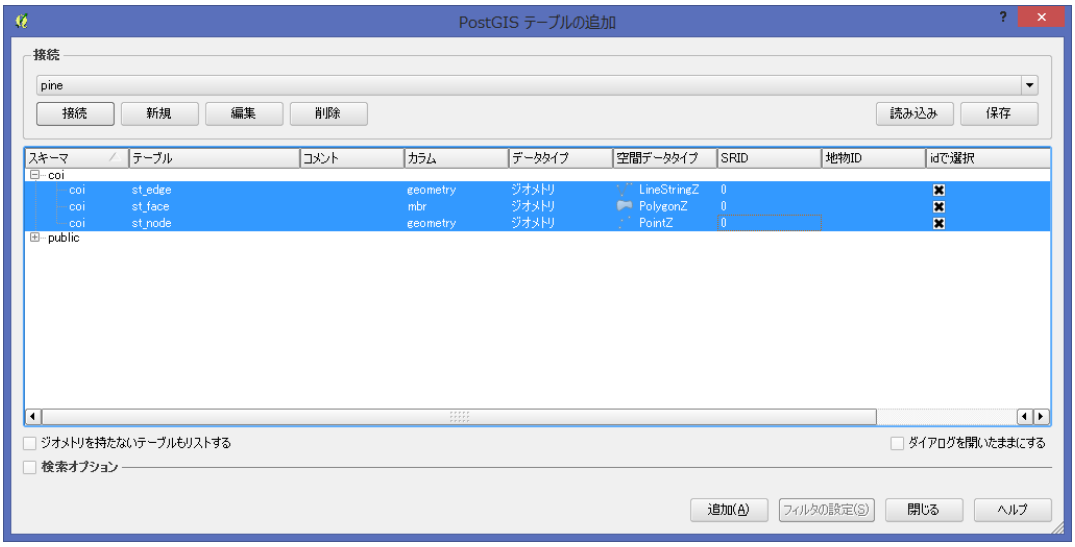
4. データベースへ接続するためのユーザ名とパスワードを入力します．



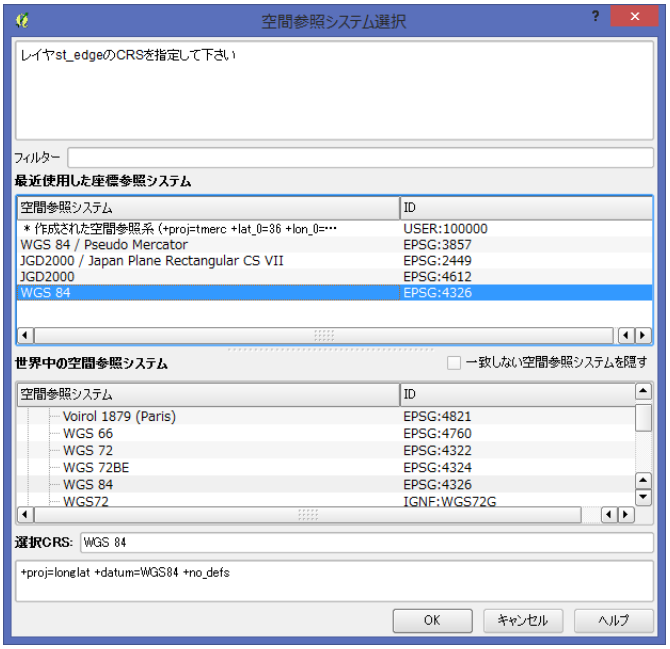
5. データベース内にある空間データ型を含んだテーブルが一覧表示されます．表示したいテーブルをすべて選択します．なお，警告マークがついているテーブルは，選択する前に SRID の列に数値を指定する必要があります．（わからない場合は 0 を指定します）



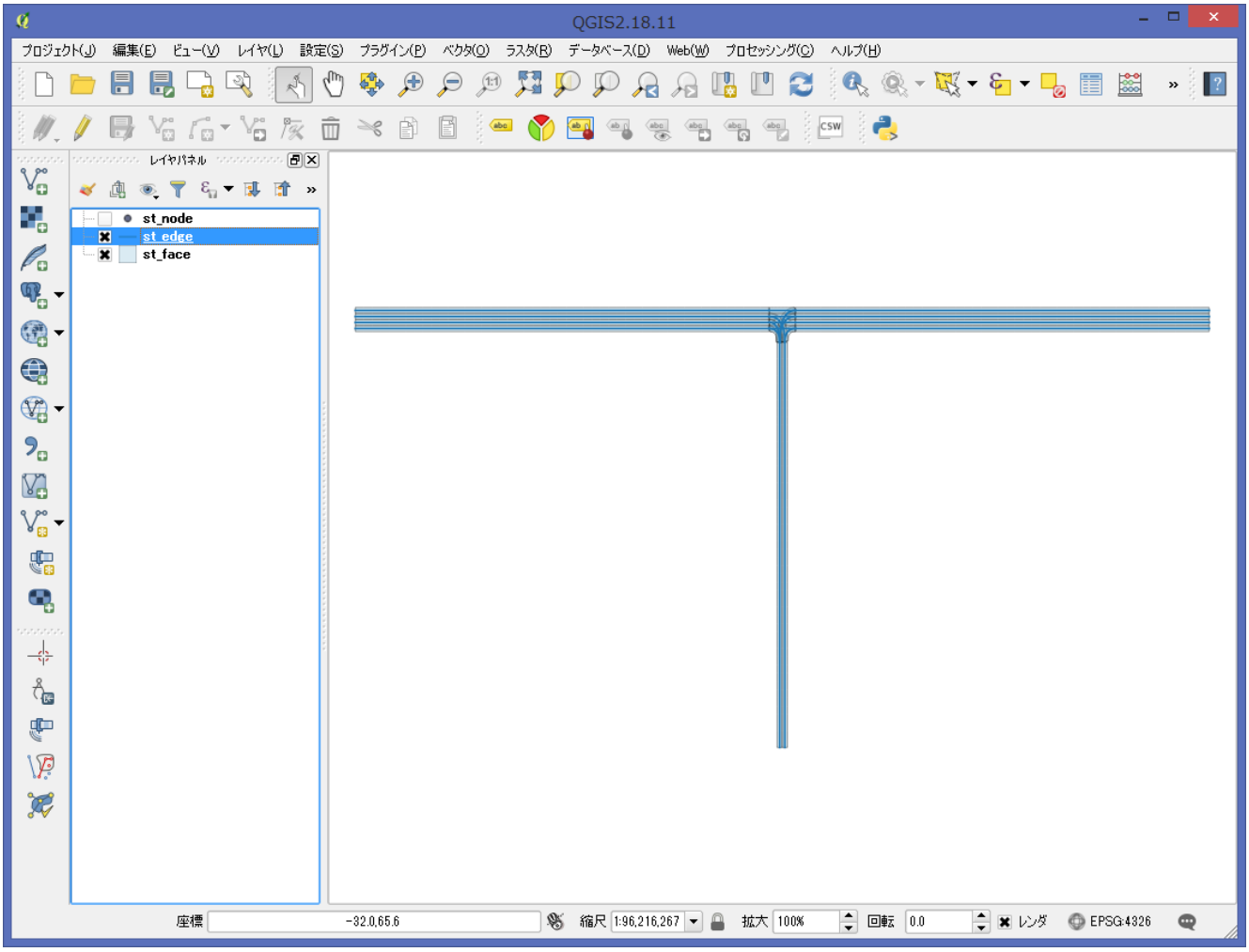
6. テーブルを選択した状態で，「追加(A)」ボタンを押します．



7. 下記の画面が表示される場合は，デフォルトの「WGS 84」を選んで「OK」を押します．



8. 選択したテーブルに格納されている空間データが表示されます.



C プログラムから PostgreSQL への問合せと結果の取得

C 言語で PostgreSQL へ接続する場合は，Libpq を用います.

Point, Line, Polygon などの空間オブジェクト（Geometry 型）は，DB から読み出した段階では hex WKB(Well-Known Binary)形式になっていますので，別のライブラリ(Geos (libgeos)や Lightweight Geometry (liblwgeom)など)を使って解析する必要があります.

サンプルコード (libpq-sample.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <libpq-fe.h>
#include <liblwgeom.h>

int main(void)
{
   PGconn *conn;            /* connection     */
   PGresult *res;           /* query result */
   int nFields, i, j;
   char *value;
   LWGEOM *g;

   /* connection information */
   const char conninfo[] = "host=127.0.0.1 port=5432 user=mirai dbname=mirai_db password=mirai";

   /* establish a connection to postgresql server */
   conn = PQconnectdb(conninfo);

   if(PQstatus(conn) != CONNECTION_OK){   /* error check */
      fprintf(stderr, "Connection to database failed: %s¥n", PQerrorMessage(conn));
      exit(EXIT_FAILURE);
   }

   /* execute a query */
   res = PQexec(conn, "SELECT node_id, geometry FROM st_node limit 10");

   if(PQresultStatus(res) != PGRES_TUPLES_OK){   /* error check */
      fprintf(stderr, "Query execution failed: %s¥n", PQerrorMessage(conn));
      PQclear(res);
      PQfinish(conn);
      exit(EXIT_FAILURE);
   }
```

```
    /* show column names of the query result */
    nFields = PQnfields(res); /* the number of columns (fields) */
    for(j=0; j < nFields; j++){
        printf("%-20s", PQfname(res, j));
    }
    printf("¥n");

    /* show each row of the query result */
    for(i=0; i < PQntuples(res); i++){

        /* show each column of the current row */
        for(j=0; j < nFields; j++){
            value = PQgetvalue(res, i, j);

            /* the first column (node_id) stores an integer data type */
            if(j == 0){
                printf("%-20s", value);
            }
            /* the second column (geometry) stores a geometry data type (hex Well-Known Binary) */
            else if(j == 1){
                g = lwgeom_from_hexwkb(value, LW_PARSER_CHECK_NONE);
                printf("%-20s", lwgeom_to_ewkt(g));    /* ewkt: extended well-known text */
            }
        }
        printf("¥n");
    }

    /* reset query result */
    PQclear(res);

    /* close the connection */
    PQfinish(conn);

    return 0;
}
```

サンプルコードのコンパイル

ファイルをコンパイルする場合は，include パスと library パスの指定が必要です.

```
% cc libpq-sample.c -I/usr/include/postgresql -L/usr/lib -lpq –llwgeom
```

Java プログラムから PostgreSQL への問合せと結果の取得

Java プログラムから PostgreSQL へ問合せを送り, 結果を取得する場合は, JDBC (Java Database Connectivity) の標準 API を使用します.
ただし，PostGIS が提供する空間データ型だけは標準ではありませんので，追加で Java bindings for postgis のライブラリを導入する必要があります．https://github.com/postgis/postgis-java/

Ubuntu の場合
% sudo　apt-get　install　–y　libpostgis-java

プログラムコードのサンプルは下記にあります.
http://postgis.net/docs/manual-2.3/ch06.html#idm2893

参考文献

(1) PostgreSQL 9.5.3 Document
（英語版）https://www.postgresql.org/docs/9.5/static/index.html
（日本語版）http://www.postgresql.jp/document/9.5/html/index.html
(2) PostGIS 2.2dev Manual
（英語版）http://postgis.net/docs/manual-2.2/
（日本語版）http://cse.naro.affrc.go.jp/yellow/pgisman/2.0.0/postgis.html
(3) Liblwgeom
http://postgis.net/docs/doxygen/2.2/da/de7/liblwgeom_8h.html
(4) GEOS – Geometry Engine Open Source
https://trac.osgeo.org/geos/