# IBM i World 2023

**IBM i コンテンツ　（2023年7月版）**

## 新登場！ IBM Watson機能を統合したIBM i （ Db2 for i ）による地理空間分析機能のご紹介（後編：実践編）

日本アイ・ビー・エム株式会社
テクノロジー事業本部
IBM Powerテクニカルセールス
澤田英寿

# IBM Watson AIをIBM iに統合し、地理空間分析が可能になりました。（後編：実践編）

## 目次

1. 地理空間列を含むテーブルを作成してみる

2. 地理空間を公開データを使って分析してみよう
   1) 公開データを取得する
   2) 公開データを使って分析する
   3) 地理分析データを地図上に展開してみよう

3. 補足情報

# 1. 地理空間列を含むテーブルを作成してみる

- IBM Db2 for iに組み込まれた、地理空間分析の機能についての概要は、
  下記の前編をご覧ください。
  https://www.jbcc.co.jp/products/files/ibmpowercolumn_202306.pdf

- 後編では、地理空間分析のイメージをつかむために、地理空間列を含むテーブルを作成して、簡単な分析を実施していきます。

- プログラム作成などは必要ありません。
  ACS（IBM i Access Client Solutions）を使って、SQLコマンドのみで実施します。

- 前提条件として、IBM i 7.5 (SF99950 レベル3以上）かIBM I 7.4(SF99704 レベル23以上）が必要です。また、Java環境が必要になります。

# 解説：

・後編では地理空間分析のイメージをつかむために、地理空間列を含むテーブルを作成してみます。

・実行前提環境
地理空間分析のDb2 for iの前提要件は、
IBM i 7.5 SF99950 レベル3　および、IBM i 7.4 SF99704 レベル23以上ですが、
地理空間機能は Java によって提供される機能を使用するため、以下の条件が必要です。

1. ジョブ CCSID を 65535 にすることはできません。
2. PASE がインストールされ、動作する必要があります。
   CHKPRDOPT PRDID(5770SS1) OPTION(33) コマンド　でPASE がインストールされていることを確認するために使用できます。
3. 地理空間関数は、JAVA 関数を介して実装され、常にデフォルトの活動化グループ、ACTGRP(*DFTACTGRP)で稼働します。

# 1. 地理空間列を含むテーブルを作成してみる

1）地理空間列を含むテーブルを作成します。

・ファイル名は、GEOTEST1　（ライブラリーはGEODATA）とします。

・フィールド名は　ID とSTATION(駅名）,LOCATION(経度・緯度）とします。

```
CREATE TABLE GEODATA.GEOTEST（ID INT , STATION CHAR(8) , STATIONJ VARCHAR(30) ,
LOCATION QSYS2.ST_POINT)
```

2）下記のコマンドで、東京駅、名古屋駅、大阪駅の座標データを挿入します。

```
INSERT INTO GEODATA.GEOTEST(ID,STATION, STATIONJ, LOCATION) VALUES(10, 'TOKYO','東京駅' ,
QSYS2.ST_POINT(139.767125, 35.681236))

INSERT INTO GEODATA.GEOTEST(ID, STATION, STATIONJ, LOCATION) VALUES(20 , 'NAGOYA', '名古屋駅' ,
QSYS2.ST_POINT(136.881537,35.170915 ))

INSERT INTO GEODATA.GEOTEST(ID, STATION, STATIONJ, LOCATION) VALUES(30 , 'OSAKA', '大阪駅' ,
QSYS2.ST_POINT(135.495951, 34.702485))
```

3）入力したデータを確認します。

```
SELECT ID,STATION,STATIONJ, ST_ASTEXT(LOCATION) AS POINT FROM GEODATA.GEOTEST;
```

右記のように表示されす。

| ID | STATION | STATIONJ | POINT |
|---|---|---|---|
| 10 | TOKYO | 東京駅 | POINT (139.767125 35.681236) |
| 20 | NAGOYA | 名古屋駅 | POINT (136.88153699999998 35.170915) |
| 30 | OSAKA | 大阪駅 | POINT (135.495951 34.702484999999996) |

解説：

・地理情報については、ジオコーディングのサイトである
https://www.geocoding.jp/　を利用して取得しました。

東京駅　　経度: **139.767125**　緯度: **35.681236**
名古屋駅　経度: **136.881537**　緯度: **35.170915**
大阪駅　　経度: **135.495951**　緯度: **34.702485**

・座標のLOCATION列の属性は、ST_POINT(経度,緯度）で表します。経度、緯度のフィールドの単位は、double(倍精度）になります。

・座標データは、デフォルトでは、WKB（Well-known binary)で表示されます。
座標を、テキストで表示するために、ST_ASTEXT 関数を用います。この関数は対応する事前割り当てテキスト(WKT) 形式 を戻します。

# 1. 地理空間列を含むテーブルを作成してみる

4) 東京駅から、名古屋駅、大阪駅までの距離を求めてみましょう。

・ST_DISTANCE関数で、直線距離を表示できます。

・東京駅の経度は、139.767125　緯度は、 35.681236 です。

SQLコマンドは下記になります。

```
SELECT STATIONJ ,QSYS2.ST_ASTEXT(LOCATION) AS point1,
QSYS2.ST_DISTANCE(ST_POINT(139.767125,35.681236) , LOCATION) as distance
FROM GEODATA.GEOTEST;
```

結果は下記になります。DISTANCE（距離）の単位はメートルです。
東京ー名古屋間は、267.8km　　東京ー大阪間は、403.5kmになります。

| STATIONJ | POINT1 | DISTANCE |
|---|---|---|
| 東京駅 | POINT (139.767125 35.681236) | 0.0 |
| 名古屋駅 | POINT (136.88153699999998 35.170915) | 267834.92746084294 |
| 大阪駅 | POINT (135.495951 34.702484999999996) | 403509.0450347522 |

解説：

・東京駅から、名古屋駅、大阪駅までの距離を求めます。最短距離を求めるには、
ST_DISTANCE関数が利用できます。

・ST_DISTANCE 関数は、入力パラメーターとして 2 つの形状オブジェクトを取り、
1 番目の形状の任意のポイントから 、2 番目の形状の任意のポイントまでの最短距離 (メートル) を戻します。

・google MAPで概算の直線距離を測定できます。東京駅ー名古屋駅、東京駅ー大阪駅間は、
下記のような距離になりました。


# 1. 地理空間列を含むテーブルを作成してみる

5）前編でご紹介した、ニューヨークの「自由の女神」までの距離を求めてみましょう。

下記のコマンドで、駅からの距離を算出できます。

```
SELECT STATION ,QSYS2.ST_ASTEXT(LOCATION) AS point1,
QSYS2.ST_DISTANCE(ST_POINT(-74.0446, 40.6893) , LOCATION) /1000  as distance
FROM GEODATA.GEOTEST1;
```

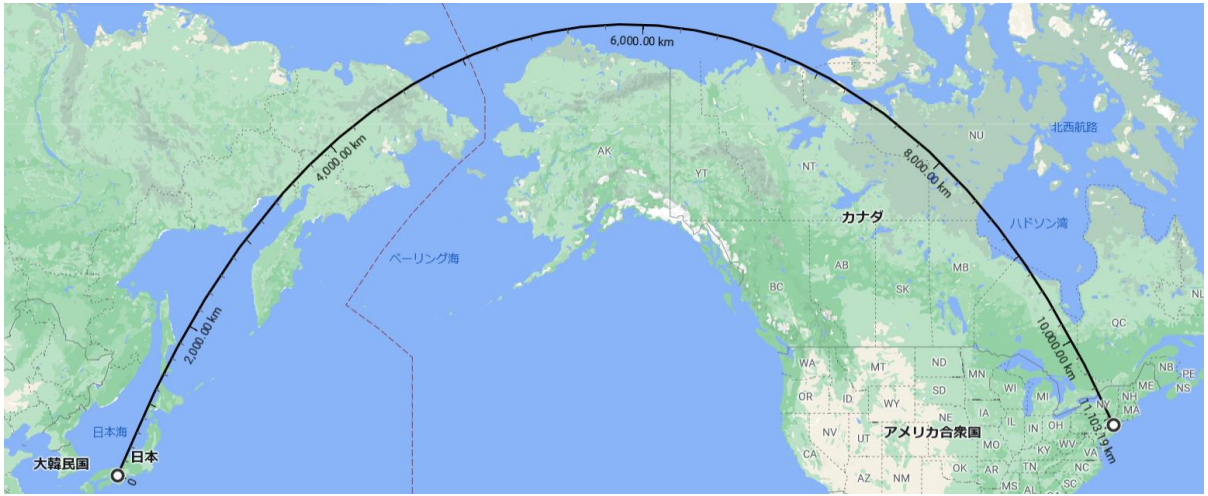
結果は下記になります。DISTANCE（距離）の単位はデフォルトはメートル単位です。
1000で割って、キロメートルで算出しています。
自由の女神ー東京駅間は、10,859.7km
自由の女神東ー名古屋間は、11,018.9km　　自由の女神ー大阪間は、11,115.9kmになります。

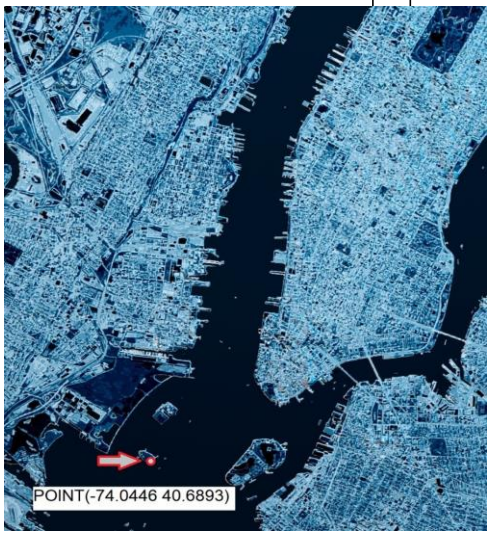
| STATION | POINT1 | DISTANCE |
|---|---|---|
| TOKYO | POINT (139.767125 35.681236) | 10859.742096516906 |
| NAGOYA | POINT (136.88153699999998 35.170915) | 11018.908649291503 |
| OSAKA | POINT (135.495951 34.702484999999996) | 11115.924868116519 |

# 解説：

・自由の女神の経度、緯度は、(-74.0446 40.6893)です。

・google MAPでも距離を測定できます。
例えば、自由の女神ー大阪駅は、地図上では、
下記のような直線距離になります。



# 1. 地理空間列を含むテーブルを作成してみる

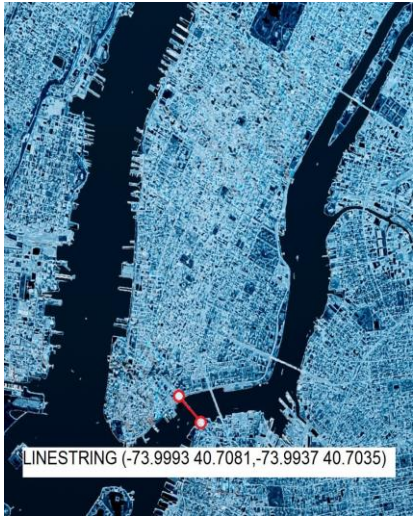
6）2地点間を距離であれば、データベースを作成しないで求めることができます。
また、形状の違った構造物の距離を求めることもできます。
ブルックリンブリッジ（橋）とセントラルパーク（公園）間の距離を求めてみましょう。

・橋のような構造物は、線（ST_LINESTRING）で表すことができます。
・セントラルパークのような公園は、境界を表す、ポリゴン（ST_POLYGON）で表すことができます。
・下記のようなコマンドで、線と境界の物体の距離を直接算出できます。

```
SELECT    QSYS2.ST_DISTANCE(QSYS2.ST_LINESTRING('linestring(-73.9993 40.7081,-73.9937 40.7035)'),
QSYS2.ST_POLYGON('polygon((-73.9580 40.8005,  -73.9813 40.7681, -73.9731 40.7647,   -73.9494
40.7967,-73.9580 40.8005)) ') ) AS DISTANCE
FROM sysibm.sysdummy1;
```

結果は下記になります。DISTANCE（距離）の単位はデフォルトはメートル単位です。
ブルックリンブリッジーセントラルパーク間の直線距離は、6,677m でした。

解説：




・ブルックリンブリッジは、線で表せます。
LINESTRING (-73.9993 40.7081,-73.9937 40.7035)　になりま

・セントラルパークのような公園は、ポリゴンで表せます。
POLYGON ((-73.9580 40.8005,
-73.9813 40.7681, -73.9731 40.7647,
-73.9494 40.7967, -73.9580 40.8005))

・google MAPでも距離を測定できます。ブルックリンブリッジーセントラルパーク間は下記のような直線距離になります。


# 2. 地理空間を公開データを使って分析してみよう

## 1）公開データを取得する（データの表示）

・様々な地理空間のデータがインターネットで公開されています。
・ここでは、JR山手線の駅データがJSON形式で公開されていますので、取得します。

駅データは、ACSから、SQLコマンドで取得できます。下記は表示のみの場合です。

```
 Select * from JSON_TABLE(
SYSTOOLS.HTTPGETCLOB(
'https://express.heartrails.com/api/json?method=getStations&line=JR%E5%B1%B1%E6%89%8B%E7%B7%9A',null),
'$.response.station[*]'
COLUMNS(
    STATIONJ VARGRAPHIC(10) PATH '$.name',
    x  DOUBLE   PATH '$.x',
    y  DOUBLE   PATH '$.y'
  )
  ) AS x;
```

| NAME | X | Y |
|---|---|---|
| 品川 | 139.738999 | 35.62876 |
| 大崎 | 139.728439 | 35.619772 |
| 五反田 | 139.723822 | 35.625974 |
| 目黒 | 139.715775 | 35.633923 |
| 恵比寿 | 139.71007 | 35.646684 |
| 渋谷 | 139.701238 | 35.658871 |
| 原宿 | 139.702592 | 35.670646 |
| 代々木 | 139.702042 | 35.683061 |
| 新宿 | 139.700464 | 35.689729 |
| 新大久保 | 139.700261 | 35.700875 |
| 高田馬場 | 139.703715 | 35.712677 |
| 目白 | 139.706228 | 35.720476 |
| 池袋 | 139.711085 | 35.730256 |
| 大塚 | 139.728584 | 35.731412 |
| 巣鴨 | 139.739303 | 35.733445 |
| 駒込 | 139.748053 | 35.736825 |
| 田端 | 139.761229 | 35.737781 |
| 西日暮里 | 139.766857 | 35.731954 |
| 日暮里 | 139.771287 | 35.727908 |
| 鶯谷 | 139.778015 | 35.721484 |
| 上野 | 139.777043 | 35.71379 |

# 解説：

・地理空間の様々なデータが、インターネットサイトで無償公開されています。
例えば、国内で有名なのは、国土交通省の国土数値情報データ（https://nlftp.mlit.go.jp/ksj/ )です。

・ここでは、HeartRails Expressというサイトから、地理空間データを取得します。
このサイトにあるAPIを使用して、 路線／駅名／最寄駅データなど様々な駅関連のデータを取得できます。
https://express.heartrails.com/api.html

・JR山手線の駅情報は、下記のURLで提供されています。
http://express.heartrails.com/api/json?method=getStations&line=JR%E5%B1%B1%E6%89%8B%E7%B7%9A
（JR%から%9Aまでの文字は、16進数で、「JR山手線」を表します）

・右記の項目が提供されています。

・ここでは右の項目のnameとx,yのみ使用します

| フィールド | 説明 |
|---|---|
| response | 駅の情報の一覧 |
| station | 駅の情報 |
| name | 駅名 |
| prev | 前の駅名 （始発駅の場合は null） |
| next | 次の駅名 （終着駅の場合は null） |
| x | 駅の経度 （世界測地系） |
| y | 駅の緯度 （世界測地系） |
| postal | 駅の郵便番号 |
| prefecture | 駅の存在する都道府県名 |
| line | 駅の存在する路線名 |

# 解説（続き）：

・このSQLコマンドは、外部公開されているRESTAPIからデータを取得する方法です。
下記資料（P38からが、[IBM iから外部のRESTAPIの使用]になります）も参照してください。
https://www.jbcc.co.jp/products/files/ibmpowercolumn20211026.pdf

・ここでは、Db2 for i の新しいHTTP機能を使用しています。
SYSTOOLS.HTTPGETCLOB関数で、RESTAPIにアクセスします。

・JSONテーブルから、データを抽出する基本構文は下記です。

```
SELECT *FROM JSON_TABLE (
                                JSON形式データ,
                                'SQL JSON パス形式'
                                COLUMNS (カラム名　データタイプ　PATH カラムパス・・・・・
                                                                )
                                        ) AS X
```

# 2. 地理空間を公開データを使って分析してみよう

## 1）公開データを取得する（公開データのIBM iデータベースへの取り込み）

・IBM iのデータベースへ取り込みします。
・取り込み用のデータベースを作成します。（STATIONPという名称にします）
　フィールド名は、STATIONJ（駅名）　X（経度）　Y（緯度）

```
CREATE TABLE GEODATA.STATIONP（STATIONJ VARGRAPHIC(10) ,X DOUBLE, Y DOUBLE);
```

次に下記のSQLコマンドでJSONデータをSTATIONPファイルに挿入します。

```
INSERT INTO GEODATA.STATIONP
Select * from JSON_TABLE(
SYSTOOLS.HTTPGETCLOB(
'https://express.heartrails.com/api/json?method=getStations&line=JR%E5%B1%B1%E6%89%8B%E7%B7%9A',null),
 '$.response.station[*]'
COLUMNS(
    STATIONJ VARGRAPHIC(10) PATH '$.name',
    x  DOUBLE   PATH '$.x',
    y  DOUBLE   PATH '$.y'
  )
  ) AS x;
```

# 2. 地理空間を公開データを使って分析してみよう

## 1）公開データを取得する（公開データのIBM iデータベースへの取り込み：続き）

とりこんだデータベースを表示してみよう。下記の30レコードが挿入されました。（山手線は30駅）

**SELECT * FROM GEODATA.STATIONP ;**

| STATIONJ | X | Y |
|---|---|---|
| 品川 | 139.738999 | 35.62876 |
| 大崎 | 139.728439 | 35.619772 |
| 五反田 | 139.723822 | 35.625974 |
| 目黒 | 139.715775 | 35.633923 |
| 恵比寿 | 139.71007 | 35.646684 |
| 渋谷 | 139.701238 | 35.658871 |
| 原宿 | 139.702592 | 35.670646 |
| 代々木 | 139.702042 | 35.683061 |
| 新宿 | 139.700464 | 35.689729 |
| 新大久保 | 139.700261 | 35.700875 |
| 高田馬場 | 139.703715 | 35.712677 |
| 目白 | 139.706228 | 35.720476 |
| 池袋 | 139.711085 | 35.730256 |
| 大塚 | 139.728584 | 35.731412 |
| 巣鴨 | 139.739303 | 35.733445 |
| 駒込 | 139.748053 | 35.736825 |
| 田端 | 139.761229 | 35.737781 |
| 西日暮里 | 139.766857 | 35.731954 |
| 日暮里 | 139.771287 | 35.727908 |
| 鶯谷 | 139.778015 | 35.721484 |
| 上野 | 139.777043 | 35.71379 |
| 御徒町 | 139.774727 | 35.707282 |
| 秋葉原 | 139.773288 | 35.698619 |
| 神田 | 139.770641 | 35.691173 |
| 東京 | 139.766103 | 35.681391 |
| 有楽町 | 139.763806 | 35.675441 |
| 新橋 | 139.758587 | 35.666195 |
| 浜松町 | 139.757135 | 35.655391 |
| 田町 | 139.747575 | 35.645737 |
| 高輪ゲートウェイ | 139.740651 | 35.635476 |

# 2. 地理空間を公開データを使って分析してみよう

## 1）公開データを取得する（取り込んだデータを座標データに変換する）

・とりこんだデータは、経度、緯度がX、Yのdoubleのフィールド形式になっているので、
地理空間データ分析するために、地理空間列「ST_POINT」のフィールドに変換する必要があります。

・STATIONP表に、LOCATION(ST_POINT）列を追加します。

```
ALTER TABLE GEODATA.STATIONP  ADD COLUMN LOCATION QSYS2.ST_POINT;
```

・LOCATIONに座標を追加します。

```
UPDATE  GEODATA.STATIONP SET LOCATION = QSYS2.ST_POINT(X,Y) ;
```

・地理空間列のデータを確認します。

```
SELECT STATIONJ, QSYS2.ST_ASTEXT(LOCATION）AS POINT FROM  GEODATA.STATIONP;
```

| STATIONJ | POINT |
|---|---|
| 品川 | POINT (139.738999 35.62876) |
| 大崎 | POINT (139.72843899999998 35.619772) |
| 五反田 | POINT (139.72382199999998 35.625974) |
| 目黒 | POINT (139.715775 35.633922999999996) |
| 恵比寿 | POINT (139.71007 35.646684) |
| 渋谷 | POINT (139.701238 35.658871) |
| 原宿 | POINT (139.70259199999998 35.670646) |
| 代々木 | POINT (139.702042 35.683060999999995) |
| 新宿 | POINT (139.70046399999998 35.689729) |
| 新大久保 | POINT (139.70026099999998 35.700874999999996) |
| 高田馬場 | POINT (139.703715 35.712677) |

# 解説

・STATIONP表には、座標列が存在していなかったので、地理空間列のLOCATION (ST_POINT)を追加しています。
・下記のように、JR山手線駅の座標が表示できました。

| STATIONJ | POINT |
|---|---|
| 品川 | POINT (139.738999 35.62876) |
| 大崎 | POINT (139.72843899999998 35.619772) |
| 五反田 | POINT (139.72382199999998 35.625974) |
| 目黒 | POINT (139.715775 35.633922999999996) |
| 恵比寿 | POINT (139.71007 35.646684) |
| 渋谷 | POINT (139.701238 35.658871) |
| 原宿 | POINT (139.70259199999998 35.670646) |
| 代々木 | POINT (139.702042 35.683060999999995) |
| 新宿 | POINT (139.70046399999998 35.689729) |
| 新大久保 | POINT (139.70026099999998 35.700874999999996) |
| 高田馬場 | POINT (139.703715 35.712677) |
| 目白 | POINT (139.70622799999998 35.720476) |
| 池袋 | POINT (139.711085 35.730256) |
| 大塚 | POINT (139.72858399999998 35.731412) |
| 巣鴨 | POINT (139.739303 35.733444999999996) |
| 駒込 | POINT (139.748053 35.736824999999996) |
| 田端 | POINT (139.761229 35.737781) |
| 西日暮里 | POINT (139.766857 35.731954) |
| 日暮里 | POINT (139.771287 35.727908) |
| 鶯谷 | POINT (139.77801499999998 35.721484) |
| 上野 | POINT (139.777043 35.713789999999996) |
| 御徒町 | POINT (139.77472699999998 35.707282) |
| 秋葉原 | POINT (139.77328799999998 35.698619) |
| 神田 | POINT (139.77064099999998 35.691173) |
| 東京 | POINT (139.766103 35.681391) |
| 有楽町 | POINT (139.763806 35.675441) |
| 新橋 | POINT (139.758587 35.666195) |
| 浜松町 | POINT (139.757135 35.655391) |
| 田町 | POINT (139.74757499999998 35.645737) |
| 高輪ゲートウェイ | POINT (139.74065099999999 35.635476) |

# 2. 地理空間を公開データを使って分析してみよう

## 2）公開データを使って分析する

・この山手線駅の公開データを使って簡単な分析をしてみましょう。
・IBM箱崎事業所からの、各駅までの直線距離が近い順に駅を並べ変えます。

SQLコマンドは下記になります。距離は1000で割ってあるのでkm単位になります。

```
SELECT STATIONJ ,QSYS2.ST_DISTANCE(ST_POINT(139.7869482,35.6783300) , LOCATION)/1000   as distance_Km
        FROM GEODATA.STATIONP ORDER BY distance_Km;
```

| STATIONJ | DISTANCE_KM |
|---|---|
| 東京 | 1.915431673473014 |
| 神田 | 2.05376885167704 |
| 有楽町 | 2.1172277018257355 |
| 秋葉原 | 2.574189732520763 |
| 新橋 | 2.898750229702918 6 |
| 御徒町 | 3.4070516002900346 |
| 浜松町 | 3.7135358296006693 |
| 上野 | 4.047682759812638 |
| 鶯谷 | 4.871284585907152 |
| 田町 | 5.083809103067737 |
| 日暮里 | 5.69768122078111 |
| 西日暮里 | 6.239551121758765 |
| 高輪ゲートウェイ | 6.347680527272787 |
| 田端 | 7.014503513826955 |
| 品川 | 7.018572428485395 |
| 駒込 | 7.400150888941778 |
| 巣鴨 | 7.496101914858163 |
| 原宿 | 7.676090587357713 |
| 代々木 | 7.695463898641185 |
| 恵比寿 | 7.7945967443883895 |
| 大塚 | 7.921573949072222 |
| 新宿 | 7.922068555270166 |
| 渋谷 | 8.048283179608314 |
| 目黒 | 8.116644387671853 |
| 五反田 | 8.159239058894254 |
| 新大久保 | 8.229604178234457 |
| 大崎 | 8.396708811617573 |
| 高田馬場 | 8.440432566767095 |
| 目白 | 8.675304862223093 |
| 池袋 | 8.968879339891595 |

一番近い駅は、東京駅で、1.9km
一番遠い駅は、池袋駅で、8.97km
になります。

# 解説

・IBM箱崎事業所の経度は、 **139.7869482**　緯度は、 **35.6783300** です。

・IBM箱崎本社からのJR山手線の駅までの直線距離を算出しています。

JR山手線 路線図（フリー版）


山手線 路線図（円形／白背景版）

# 2. 地理空間を公開データを使って分析してみよう

## 3）地理分析データを地図上に展開してみよう

・この山手線駅の地理データを地図上に展開してみましょう。

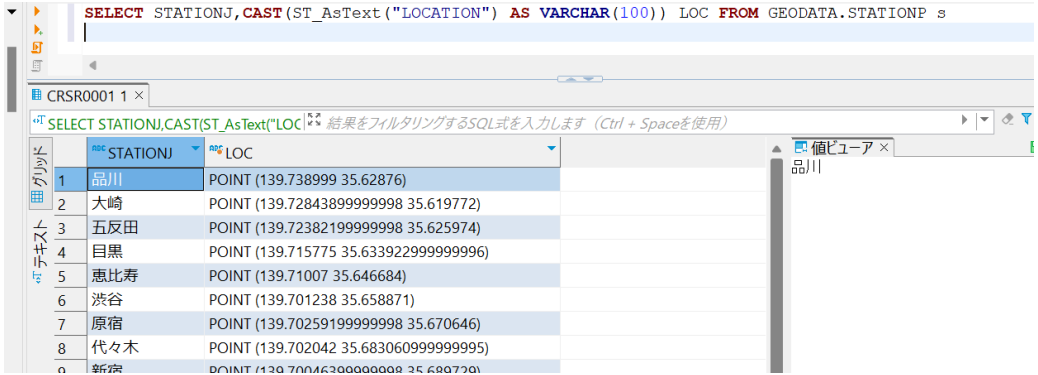
①オープンソースである、DBeaverをPC上に導入します。

②DBeaverを起動して、IBM iのデータベースを選択し、接続します。

③下記のようにSQLコマンドを入力します。先ほどの山手線駅の地理空間データを使用しています。

```
SELECT STATIONJ,CAST(ST_AsText("LOCATION") AS VARCHAR(100)) LOC FROM
GEODATA.STATIONP s
```

④下記のように表示されます。




（参考）ここでのDbeaverでの地理空間分析の方法は、下記に詳細が掲載されています。

「Db2地理空間分析: DBeaverでDb2の地理空間情報を表示してみる」

https://qiita.com/nishikyon/items/60e23324b1ec1f65deb5

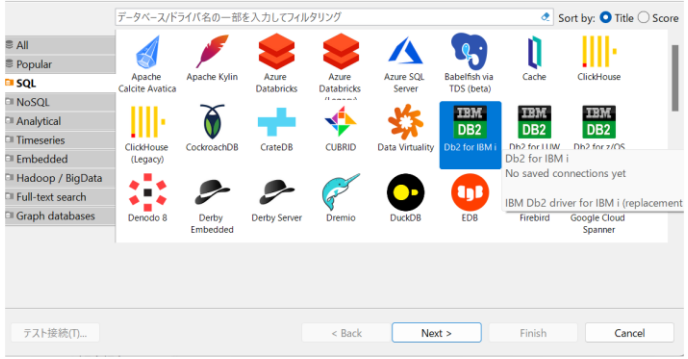
# 解説

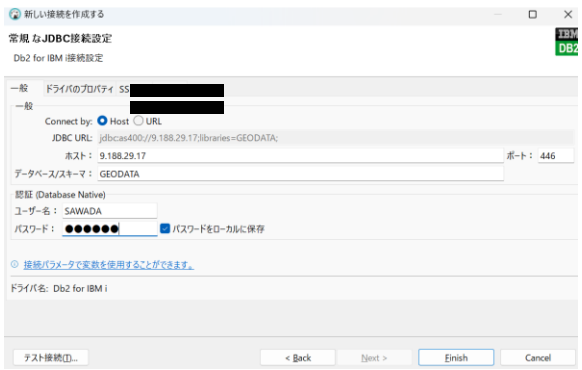
- DBeaverは、フリーライセンスで、Apache ライセンス バージョン 2 の条件に基づいて、再配布したり変更したりすることができます

- DBeaverはマルチプラットフォーム対応で、IBM iを含む多数のデータベースに接続できます。地図と連動する機能を持っています。

- 現在（2023/7/5)の最新版は、DBeaver Community 23.1.1です。
  下記からWindows版、mac版、Linux版をダウンロードできます。
  https://dbeaver.io/download/

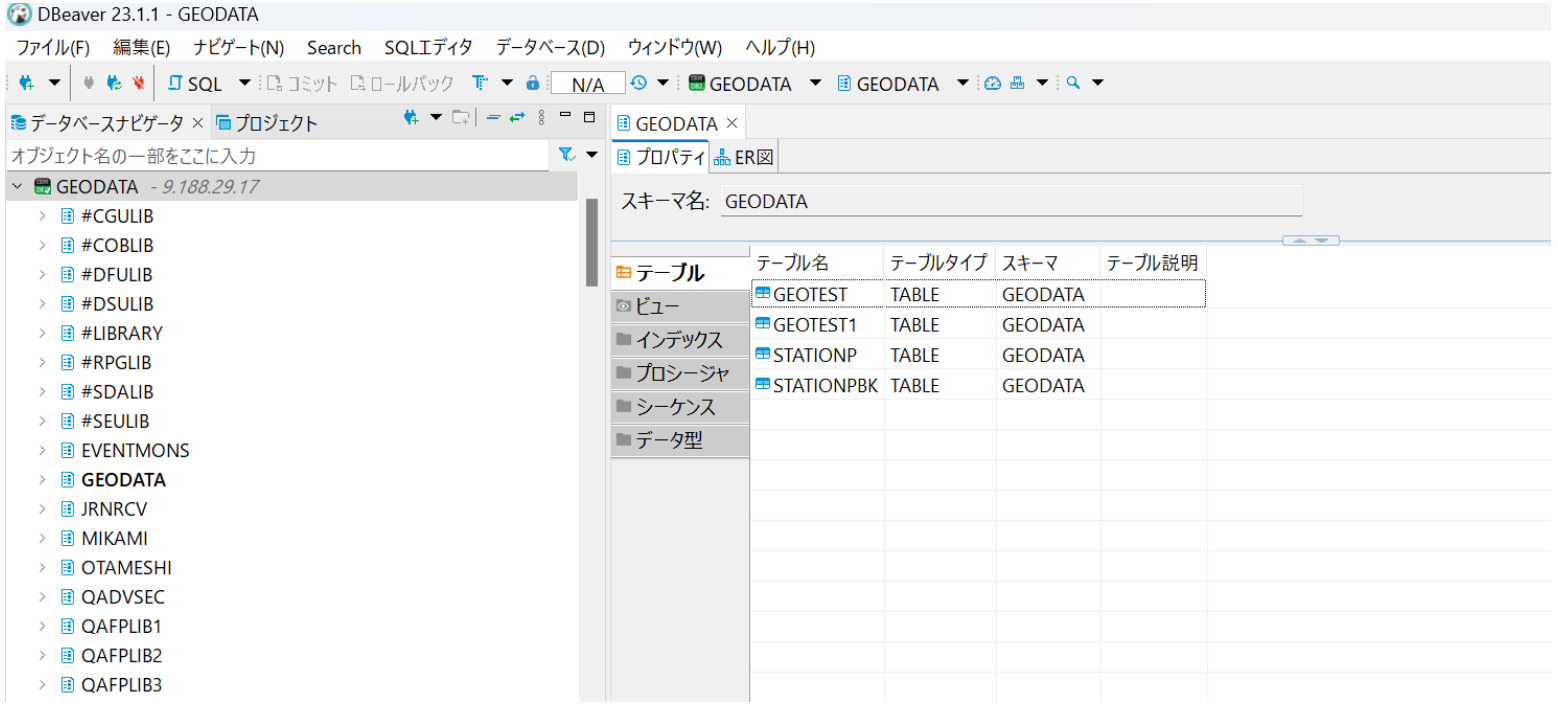
- IBM iに接続するときは、下記のように、Db2 for IBM iを選択してください。
  接続時にスキーマ（ここではGEODATA)を指定しておきます。



# 解説（続き）

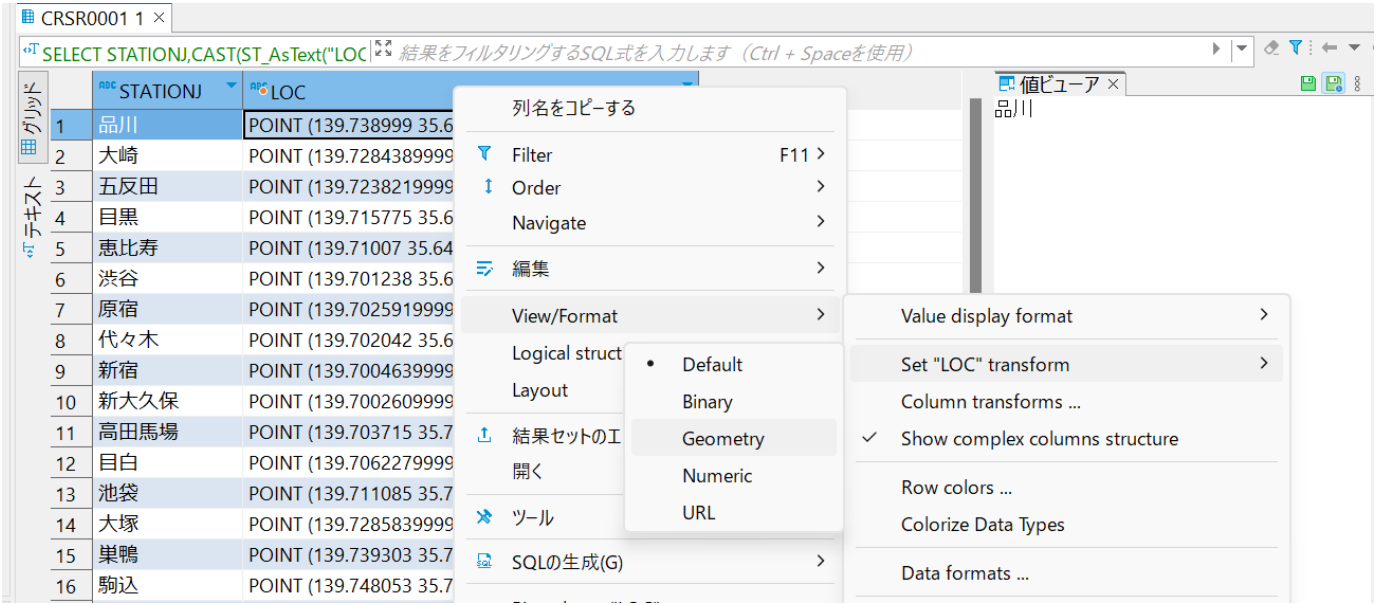
・DBeaverで、IBM i に接続すると最初の画面は下記になります。



# 2. 地理空間を公開データを使って分析してみよう

## 3）地理分析データを地図上に展開してみよう（続き）

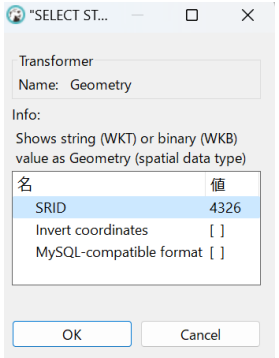
⑤下記のように、[LOC]列を右クリックして、[view/format]→Set”LOC”transform →[Geometry]の準に選します。これで、この列が地理空間データとして認識されます。



# 2. 地理空間を公開データを使って分析してみよう

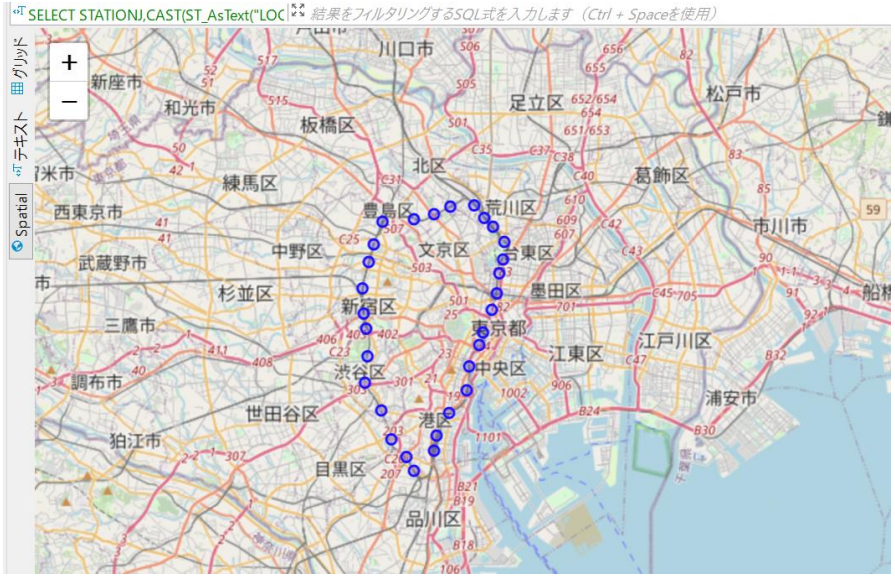
## 3）地理分析データを地図上に展開してみよう（続き）

⑥下記のように表示されるのでOkを選択




⑦下記の左のタブにある、「Spacial」を選択

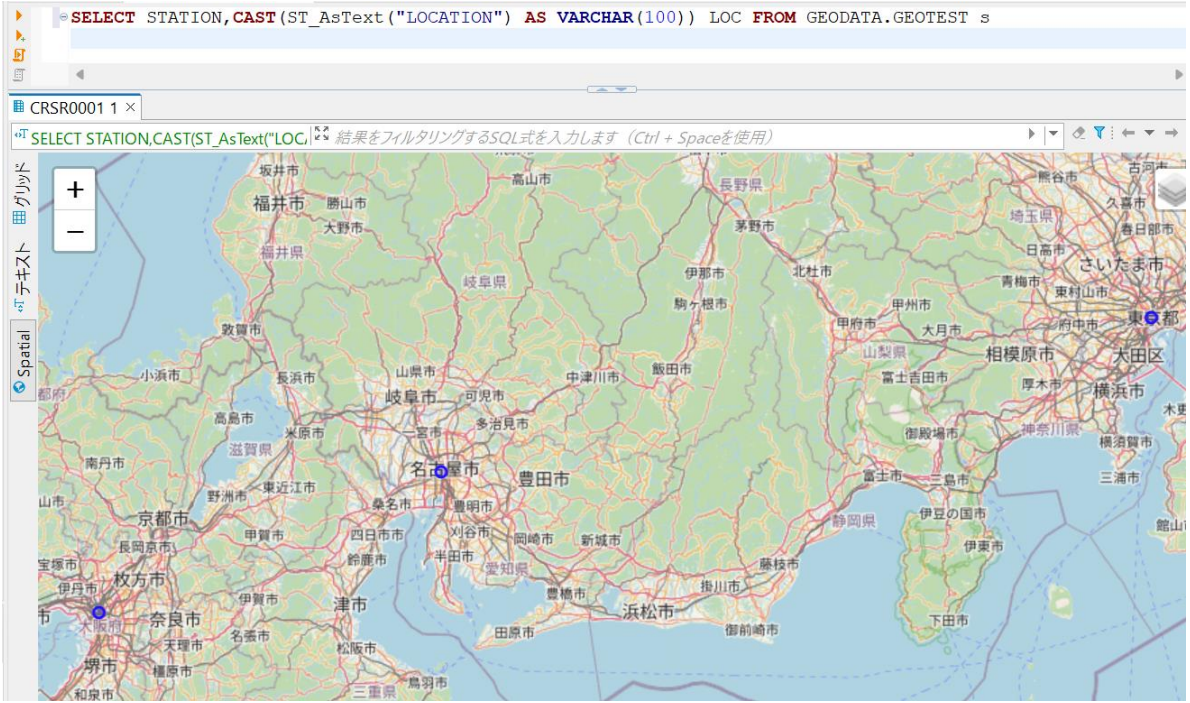

⑧下記のように地図上に、山手線の駅データがマッピングされます。拡大・縮小可能です。



# 2. 地理空間を公開データを使って分析してみよう

## 3）地理分析データを地図上に展開してみよう（続き）

⑨以前作成した、東京駅、名古屋駅、大阪駅の地理データも同様に地図上にマッピングしてみます。
駅が青色でマッピングされています。



# 3. 補足情報

(1) IBM i 7.5マニュアル「地理空間分析」
https://www.ibm.com/docs/en/i/7.5?topic=database-geospatial-analytics

(2)Db2 でシェアサイクルポートの地理分析をやってみよう！
https://ibm-developer.connpass.com/event/259583/

(3) Db2地理空間分析: DBeaverでDb2の地理空間情報を表示してみる
https://qiita.com/nishikyon/items/60e23324b1ec1f65deb5

(4) 無償で使える地理空間データ
https://www.esrij.com/gis-guide/other-dataformat/free-gis-data/

# IBM i 情報

IBM i ポータル・サイト
https://ibm.biz/ibmijapan

月イチIBM Power情報セミナー「IBM Power Salon」
https://ibm.biz/power-salon

IBM i World 2021 オンデマンド・セミナー
https://ibm.biz/iworld2021

IBM i ホワイトペーパー 2021年日本語版
https://www.ibm.com/downloads/cas/JB8AXO9V

IBM i Club (日本のIBM i ユーザー様のコミュニティー)
https://ibm.biz/ibmiclubjapan

i Magazine (IBM i 専門誌。春夏秋冬の年4回発刊)
https://www.imagazine.co.jp/

IBM i 情報 Facebook
https://www.facebook.com/iusersjapan

IBM i 研修サービス (i-ラーニング社提供)
https://www.i-learning.jp/service/it/iseries.html

Fix Central (HW・SWのFix情報提供)
https://www.ibm.com/support/fixcentral/

IBM My Notifications (IBM IDの登録 [無償] が必要)
「IBM i」「9009-41G」などPTF情報の必要な製品を選択して登録できます。
https://www.ibm.com/support/mynotifications

IBM i 7.5 技術資料
https://www.ibm.com/docs/ja/i/7.5

IBM i 各バージョンのライフサイクル
https://www.ibm.com/support/pages/release-life-cycle

IBM i 以外のSWのライフサイクル (個別検索)
https://www.ibm.com/support/pages/lifecycle/