空間データベースを用いた
沿線情報検索システム構築の試み

島根大学　総合理工学部　数理・情報システム学科

応用情報学講座　田中研究室

S063014　宇垣 貴裕

目次

第１章　序論

１．１　研究目的

これから乗るバスの沿線情報を表示することは、目的地に行くまでに途中で降りて立ち寄ることを可能にする。地図上にバスが通過する経路を表示することで、近隣観光スポットや近隣店舗を知ることができる。沿線情報を検索するに当たって今から乗る部分の路線座標を抜き出す必要があるので、今回新たに追加した機能として先行研究[1]をベースに路線座標を抜き出して、バス時刻表と共に GoogleMaps[2]上に表示するシステムの構築を行った。

また、先行研究で用いた DBMS の HiRDB はオープンソースでなく有償であり、本などの資料も少ないので、今回、DBMS をオープンソースであり、HiRDB と同様に空間データの拡張も出来る PostgreSQL[5]に変更した。

１．２　概要

前述したシステムの実装にあたって、先行研究と同様に座標データの管理や検索には DBMS を用いた。本システムでは座標データといった文字や数値以外のデータを容易に扱うことができ、地図を用いた空間検索が可能な ORDBMS を使用した。

また、先行研究では Web サーバにはオープンソースの Apache[3]、クライアントと Web サーバをつなぐ言語には Java サーブレット、サーブレットコンテナには Tomcat[4]、Web サーバと DB サーバ間の接続には JDBC ドライバ、クライアント上での地図の表示や地図上でのマーカーの表示は Google Maps API を用いており、本研究でも同様のものを使用した。

１．３　先行研究概要

先行研究[1]は GoogleMaps を用いて PC 又は GPS 携帯端末を用いた現在位置の情報と目的地の情報を元に近隣バス停の中で最適なバス時刻を検索して表示するシステムを実装している。本研究ではこの先行研究を元に路線座標を抜き出して表示するシステムの実装を行った。

図 1.3.1　先行研究での結果表示

第２章　システム概要

２．１　Web サーバ

Web サーバとは WWW システムにおいて情報送信を行うコンピュータ、または WWW による情報送信機能を持ったソフトウェアのことである。

Web サーバは、HTML 文書や画像などの情報を蓄積しておき、Web ブラウザなどのクライアントソフトウェアの要求に応じて、インターネットなどのネットワークを通じて、これらの情報を送信する役割を果たしている。

本システムでは Web サーバとして Apache2.2[3]を用い、Web サーバ・サーブレットコンテナとして Tomcat6.0[4]を用いる。Apache と Tomcat を連携させることにより、Java サーブレットや JDBC を用いた Web システムの構築をおこなっている。具体的には静的コンテンツを Apache が処理し、動的コンテンツを Tomcat が処理するといった方法をとっている。Tomcat にも Web サーバの機能は存在するが、Apache に比べ処理が遅く、細かな設定が出来ないという理由から本システムでは使用せず、Apache を用いる。

２．２　Java サーブレット

Java サーブレットとは、Web サーバ上で実行されるモジュール(部品)化された Java プログラムのことである。サーブレットを追加することにより、Web サーバの機能を拡張することができる。サーブレットは Java 言語で記述されているため、特定の OS やハードウェアに依存せずにサーブレット API を実装したあらゆる Web サーバで稼動させることが出来る。また、サーブレットと似た技術で CGI がある。CGI に対するサーブレットの利点を次に示す。

- 効率的

  CGI の場合、HTTP リクエスト毎に新たなプロセスが生成される。それに対してサーブレットは、Java 仮想マシンが唯一のプロセスとして常に動作しており、個々のリクエストはスレッドによって処理される。そのため、プロセス起動のためのオーバーヘッドが生じない。また、処理終了後もメモリ上に存在するため、複数のリクエストから簡単に利用できる。
- 開発のしやすさ

  サーブレットは HTML フォームデータの構文解析、HTTP ヘッダの読み取りと設定、クッキーの処理、セッションの管理、といった多くの基礎的機能を豊富に備えている。また、信頼性も高く再利用性にも富む。
- 強力

  複数サーブレットでのデータベース接続の共有など、資源を共有する最適化を容易に実装することができる。また、サーブレットは複数のリクエストにまたがって情報を保持するため、セッション管理や、前の処理結果のキャッシュなどの技術も簡単に実現できる。
- 可搬性

サーブレットは Java の標準 API を使用するため、一般的にプラットフォーム非依存である。

・ 安全性

CGI は通常 OS のシェルから実行するため、特殊文字（¥n など）に対して細心の注意が必要である。また、配列に対するバッファオーバフローの攻撃対象になりうる。これに対し、サーブレットはこれらの問題点がない。

２．３　JDBC

JDBC とは Java プログラムからリレーショナルデータベースにアクセスするための API のことである。SQL 言語による命令を発行してデータベースの操作を行なうことができ、データベースの種類によらない汎用性の高いプログラムを開発することが可能である。JDBC ドライバは、各 DBMS 用のドライバが必要である。PostgreSQL 用の JDBC ドライバを用いて Java サーブレットから DBMS へ SQL 文を発行して、検索と検索結果の受け取りを行っている。

２．４　Data Source

DBMS と Tomcat の連携には JDBC ドライバを用いた Data Source による接続を行っている。

Data Source とは、Tomcat に JDBC ドライバを定義することで、サーブレットが DBMS へ接続から切断までの処理を行う必要がなくなる。また、１度接続したらその接続を保持することで、DBMS への接続時間も短くなり、さらにその接続を他のサーブレットに使わせることも可能であるというメリットがある。

それに対し、Data Source と同様の処理が行える Driver Manager では、個々の Java プログラム内に JDBC ドライバを定義し、接続はそれぞれのサーブレットが行い、サーブレットが生成されるたびに、DBMS への接続から切断までの処理が行われる。

このような理由により本研究では Data Source による接続を選択した。

第３章　システム移行

３．１　ORDBMS

空間検索オブジェクトリレーショナルデータベース（ORDB）とは、従来のリレーショナルデータベース（RDB）の延長線上に位置しており、RDB にオブジェクト指向の環境を提供するものである。また、RDBMS において別々の DB に格納していた汎用データと空間データを ORDBMS では同一の DB に格納することが可能である。同一の DB 内で取り扱うことにより空間情報と文字数値データを属性データとして同一行内に関連付けることが出来る。さらに、空間情報そのものを条件として検索することも可能である。

先行研究では純国産 ORDBMS である日立 HiRDB Version 7 を用いてユーザデータやバス停情報といったデータベースの管理を行っていたが、本研究ではオープンソースの PostgreSQL[5]を用いた。HiRDB では HiRDB 空間検索プラグイン(HiRDB Spatial Search Plug-in Version 3)によって空間情報を扱っていた。本研究では PostgreSQL に空間情報を拡張する為に PostGIS[6]を用いた。これによって HiRDB と同様、Point 型、LineString 型、Polygon 型などの空間情報を扱うことができる。これによってバス停などの位置情報を比較的簡単な SQL 文の記述によって、検索することができる。

３．２ 空間データ操作

ORDB では大まかに Point 型（点）、LineString 型（線）、Polygon 型（多角形）が存在している。これらのデータを DWithin メソッドにより検索範囲に図形が完全に含まれているかを判定することができる。

なお、HiRDB の空間検索では Within メソッドを用いていたが、PostgreSQL の空間検索では Within メソッドは検索半径を指定出来ない為、検索半径の指定出来る DWithin メソッドを利用している。

空間データの表現方法として Well-Known Text（WKT）フォーマット形式がある。これは、OGC によって定義されている図形をテキストで表現する方法であり、空間検索システムや SQL などのこれをサポートしているソフトウェア同士でデータの受け渡しを容易に行うことができる。本研究で用いている PostGIS でも用いている。図 2.2.1 に WKT の例を挙げる。

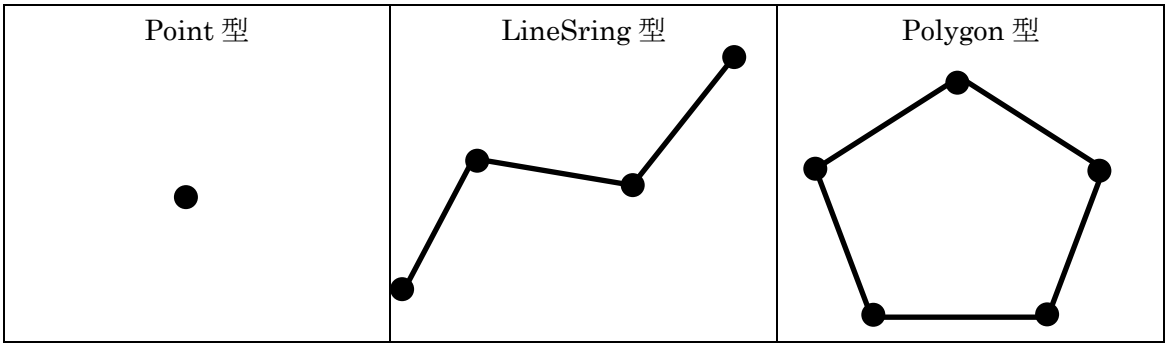


図 2.2.1　WKT 表現の例

３．２．１　DWithin メソッド

DWithin メソッドというのは検索する列にある Point、LineString、Polygon 型で格納されているデータと検索範囲の図形との空間関係をチェックして、前者が後者に完全に含まれるかどうかを判定する機能を持つ。DWithin メソッドは PostGIS 独自のものであり、HiRDB の空間検索プラグインに DWithin メソッドは存在しない。このメソッドによって、その円の中に Point 型で格納されているバス停の座標が含まれているかどうかを判定したり（図 3.2.2）、LineString 型で格納されているバス路線の中に飲食店の座標が含まれているかどうかを判定したりすることができる（図 3.2.3）。これによって現在位置から 500m 以内にあるバス停の検索やバス路線の沿線上にある飲食店の検索などを行うことができる。

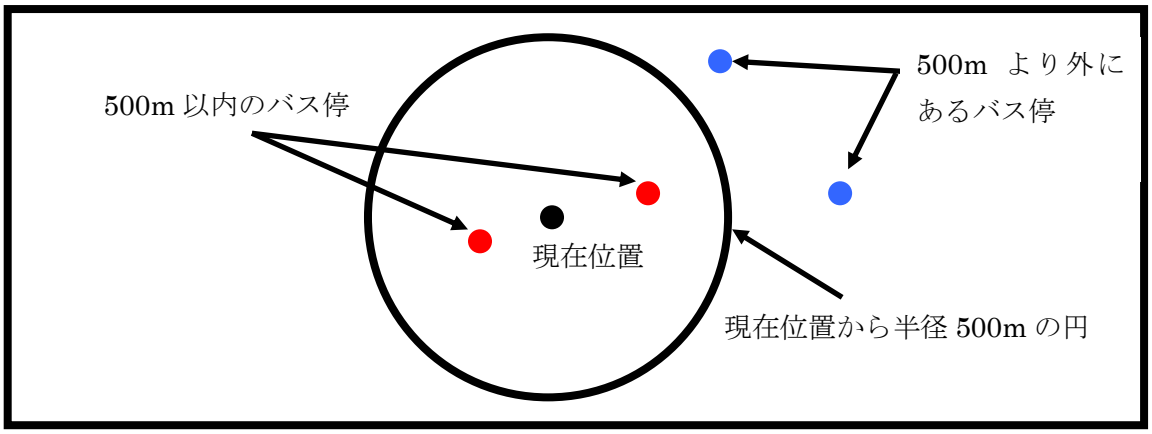


図 3.2.2　半径 500m 以内にあるバス停の検索

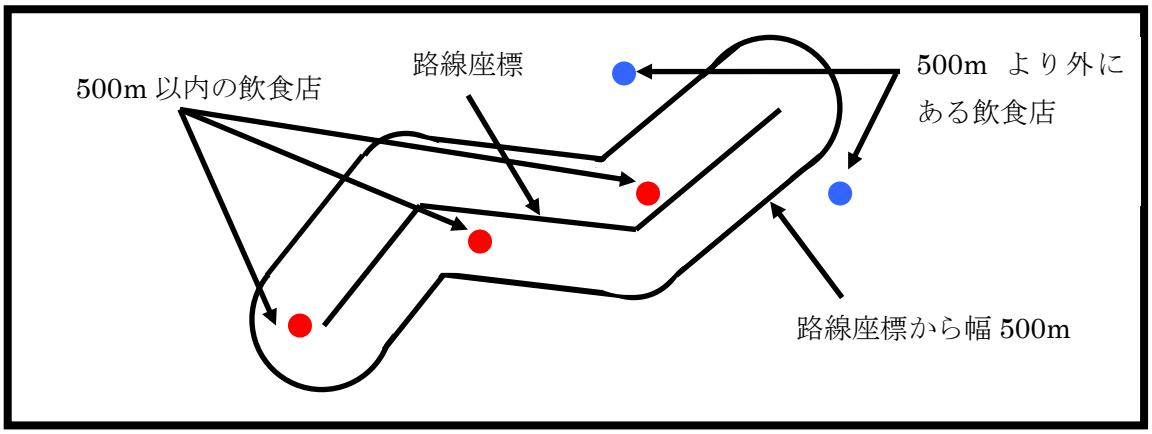


図 3.2.3　路線座標から 500m 以内にある飲食店の検索

３．３　SQL 文のシンタックスの変更

HiRDB から PostgreSQL へ DBMS を変更したことによってプログラム内の空間検索 SQL 文のシンタックスを PostGIS に対応したものに変更する必要があった。下に半径 500m 以内のバス停を検索する場合の SQL 文を例にあげる。

・HiRDB の場合

```
SELECT ID,バス停名,X 座標,Y 座標,バス会社
FROM バス停位置 13WGS
WHERE WITHIN(座標, RegionFromText('CIRCLE(" + start_x + " " + start_y + ", 5)'))
IS TRUE
```

・PostgreSQL の場合

```
SELECT ID,バス停名,X 座標,Y 座標,バス会社
FROM バス停位置 13WGS
WHERE ST_DWITHIN(座標, GeomFromText('POINT(" + start_x + " " + start_y +
")',4326),5)
= TRUE
```

上の 2 つの SQL 文は同じ意味の SQL 文だが、比べると WHERE 以下の条件部分が違っていることが分かる。先行研究を再現するにはプログラム内の空間検索 SQL 文を全て上のように修正する必要があった。なお、PostgreSQL の場合の SQL 文の 4326 というのは SRID（空間参照識別子）というもので空間参照系を指定している。

第４章　路線座標表示機能の追加

４．１　システム概要

本システムは先行研究のバス停時刻検索システムに沿線情報を検索するための準備として、ユーザーが乗車する部分のバス路線を GoogleMaps 上に表示する（図 4.1.1)。

図 4.1.1　PC(左)と携帯端末(右)での表示結果

４．２　システムの流れ

４．２．１　PC の場合

PC の場合の使用方法は、先行研究と同様、地図上から出発地と目的地をクリックし、平日か休日を選択し(①)、その情報を Web サーバに送信する(②)。Web サーバは受け取ったデータを元に JDBC を用いて SQL 文を発行し(③)、DBMS に接続(④)、出発地周辺のバス停と目的地周辺のバス停を検索し(⑤)、検索結果のバス停から路線を検索する(⑥)。この路線を元に時刻を検索し(⑦)、出発のバス停と目的のバス停の位置と時刻を取得し(⑧)、取得した情報をクライアントへ送信する(⑨)。クライアントはその情報から時刻と地図上に出発のバス停と目的のバス停と出発のバス停から目的のバス停までの路線図を表示する(⑩)。Web サーバとの通信には Ajax による非同期通信を用いて、画面遷移のない高速な処理を行っている。「次へ」をクリックすると、次の時刻を表示する。これらの流れを図 4.2.1 に示す。

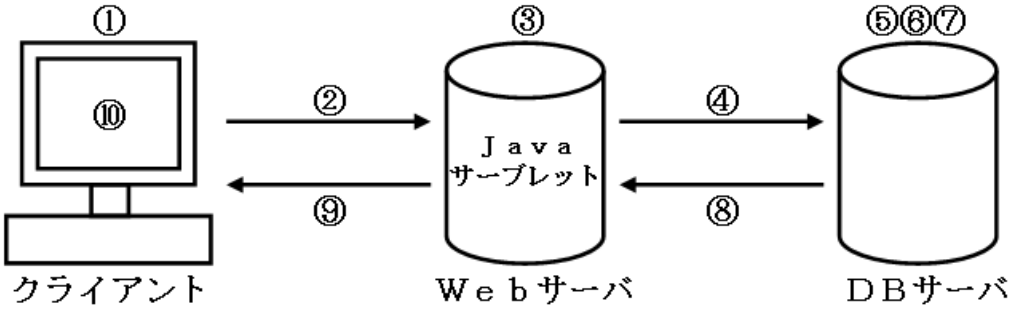


図 4.2.1　PC の場合のシステムの流れ

４．２．２　携帯端末の場合

携帯端末の場合の使用方法は、まず、GPS 又は簡易位置情報機能付きのものの場合は現在位置を取得し(①)、その情報を Web サーバに送信する(②)。Web サーバはこの情報を元に周辺地図を生成し(③)、クライアントへ送信する(④)。クライアントはその周辺地図から現在位置を選択し(⑤)、その結果を Web サーバへ送信する(⑥)。GPS 又は簡易位置情報機能がない場合は、松江市全体の地図が表示されここから現在位置を選択し(①)、その情報を Web サーバに送信する(②)。次からは GPS 又は簡易位置情報機能付きである場合もない場合も同様で、Web サーバは目的地検索のために松江市全体の地図を生成し(⑦)、クライアントへ送信する(⑧)。クライアントはこの地図から目的地を選択する(⑨)。次に日時を選択し(⑩)、それらの情報を Web サーバへ送信する(⑪)。Web サーバは送信されてきたデータを元に JDBC を用いて SQL を発行して(⑫)DBMS に接続し(⑬)、現在位置周辺のバス停と目的地周辺のバス停を検索し(⑭)、路線を検索する(⑮)。この路線を元に時刻を検索し(⑯)、Web サーバへ送信する(⑰)。その中で最適な時刻と出発のバス停、目的のバス停の位置を表示した地図をクライアントへ送信する(⑱)。クライアントは地図と時刻を取得する(⑲)。次の時刻を選択すると次の時刻を検索することができる。これらの流れを図 4.2.2 に示す。

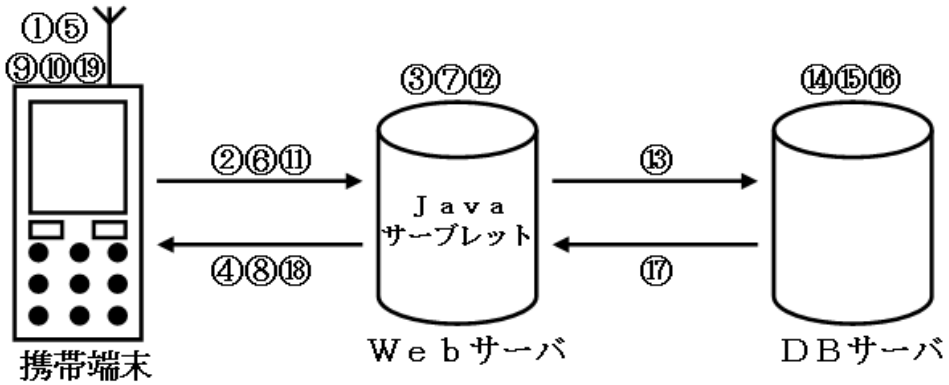


図 4.2.2　携帯端末の場合のシステムの流れ

４．３　PC での位置情報取得方法

Google Maps API を用いてクリックした位置の情報を取得することで出発地と目的地の位置情報を取得することが出来る。以下に具体的な方法としてプログラムを記述する。

Google Maps API を用いた位置情報取得方法

```
function onLoad(){
   //地図作成
   map = new GMap2(document.getElementById('map'));
   map.addControl(new GScaleControl());
   map.setCenter(center, 13);
   //クリックした位置を取得
   GEvent.addListener(map, 'click', onsMapClick);
}
//クリック処理
function onsMapClick(overlay, point){
   x=point.x;   //x に緯度格納
   y=point.y;   //y に経度格納
}
```

ここでは、GEvent.addListener(map, 'click', onsMapClick);によりクリックした位置の緯度・経度をそれぞれ point.x、point.y に格納する。その緯度・経度を x と y に格納することにより、クリックした位置から緯度・経度を取得することが出来る。

４．４　携帯端末での位置情報取得方法

携帯端末を用いた位置情報の取得方法は現在位置の取得には GPS や簡易位置情報というものがある。

GPS とは Global Positioning System（全地球測位システム）の略で、米国防総省が管理する衛星からの電波を利用し、緯度、経度、高度などを数十メートルの精度で割り出し、現在地を知らせるシステムのことである。近年、GPS はカーナビゲーションシステムや携帯電話に広く組み込まれるようになった。docomo、SoftBank、au、WILLCOM の 4 社全てが GPS 機能を提供しており、ほとんどの端末に GPS 機能が組み込まれている。

また、簡易位置情報というのは、コンテンツ側からの要求により、利用者同意の上で、端末の大まかな位置情報を送信するというものである。ただし、送信される位置情報は端末のおおよその位置を示すものであり、正確な位置とは異なり、500m～十数 km の誤差が発生する。簡易位置情報については docomo、SoftBank、au の 3 社がこの機能を提供しており、ほとんどの端末にこの機能が組み込まれている。

そこで、本システムでの現在位置取得方法として GPS と簡易位置情報の両方に対応したものを実装することとした。また、携帯会社それぞれで取得方法や URL の記述方法が若干違うのでそれぞれの会社に対応したものを作成する必要があった。

４．４．１　現在位置取得

上でも述べているように現在位置取得方法にはGPSと簡易位置情報があるが、簡易位置情報は大きな誤差生じてしまうしGPSでも誤差が生じてしまうかもしれないので、本システムではGPS又は簡易位置情報を用いて周辺地図を生成し、周辺地図から選択させる方法をとることとしている。また、GPSや簡易位置情報といった機能を搭載していないものについてはPCの時と同様、松江市全体の地図から現在位置を選択するようにしている。

なお、目的地の選択ではGPSや簡易位置情報機能を用いることが出来ないので、PCの時と同様、松江市全体の地図から目的地を選択するようにしている。

４．４．２　GPSと簡易位置情報

GPSまたは簡易位置情報の取得方法はURLにそれぞれ必要なキーワード付加することで使用することができる。Javaサーブレットに位置情報を取得させるには以下のように記述する。

・ auのGPSの場合[10][11]

```
<a href="device:gpsone?url=http://rena.cis.shimane-u.ac.jp/project02/servlet/Position&ver=1&datum=0&unit=0">au（GPS）版</a>
```

・ auの簡易位置情報の場合[10][11]

```
<a href="device:location?url=http://rena.cis.shimane-u.ac.jp/project02/servlet/Position">au（簡易位置情報）版</a>
```

という記述になる。deviceの後ろがgpsoneの場合はGPSによる測位となり、locationの場合は簡易位置情報による測位となる。ここで、は" "で囲まれたURLへのリンクを表すタグであり、"文字"が実際の画面に表示される。また、以下はパラメータの内容である。

・ ver：GPSのバージョン
・ datum：測地系(0：世界測地系WGS84、1：日本測地系TOKYO)
・ unit：緯度経度の表記方法（0：dd.mm.ss.sss度分秒表記、1：dd.ddd度表記）

以上の3つのパラメータはGPS取得の際に必要である。実際にこのURLを記述したページにアクセスし、位置情報を取得すると、以下のようなパラメータを付加した上で、そのurl=で指定したURLに移動する。

```
http://rena.cis.shimane-u.ac.jp/s033033/servlet/Position?ver=1&datum=0&unit=0&lat=+35.28.52.82&lon=+133.04.12.85&alt=51&time=20031219154915&smaj=15&smin=10&vert=25&majaa=14&fm=0
```

上記のパラメータは、

・ lat：緯度
・ lon：経度
・ alt：高度
・ time：取得日時

・ smaj：長軸半径誤差
・ smin：短軸半径誤差
・ vert：高度誤差
・ majaa：誤差楕円長軸角度
・ fm：測位方法（何を用いて測位したか）
である。

・ SoftBank の場合[10][12]

```
<a href="location:auto?url=http://rena.cis.shimane-u.ac.jp/project02/servlet/
CurrentPosition">SoftBank 版</a>
```

という記述になる。location の後ろが cell の場合は簡易位置情報による測位となり、gps の場合は GPS による測位となり、auto の場合は端末で優先された測位となる。本システムでは汎用性の高さから auto を用いている。実際にこの URL を記述したページにアクセスし、位置情報を取得すると、以下のようなパラメータを付加した上で、その url=で指定した URL に移動する。

```
http://rena.cis.shimane-u.ac.jp/project02/servlet/Position?pos=N35.28.52.82E133.04.
12.85&geo=wgs84&x-acr=3
```

上記のパラメータは、
・ pos：座標値（1/100 秒単位で度分秒表記）N 北緯、S 南緯、E 東経、W 西経
・ geo：測地系（wgs84：世界測地系、tokyo：日本測地系、itrf：ITRF 系）
・ x-acr：精度（1：簡易位置情報(300m 以上)2：GPS(50m～300m)3：GPS(50m 以内)）

・ docomo の場合[10][13]

```
<a href="http://rena.cis.shimane-u.ac.jp/project02/servlet/CurrentPosition&
com=docomo" lcs>docomo 版</a>
```

という記述になる。Lcs を付加することで GPS を取得することが出来る。実際にこの URL を記述したページにアクセスし、位置情報を取得すると、以下のようなパラメータを付加した上で、指定した URL に移動する。

```
http://rena.cis.shimane-u.ac.jp/project02/servlet/Position?
lat=+35.28.52.82&lon=+133.04.12.85&geo=wgs84&x-acc=3
```

上記のパラメータは、
・ lat：緯度（全て±dd.mm,ss.sss の度分秒表記）
・ lon：経度（全て±dd.mm,ss.sss の度分秒表記）
・ geo：測地系（wgs84：世界測地系、tokyo：日本測地系）
・ x-acc：水平誤差（3：50m 以内、2：50m 以上 300m 未満、1：300m 以上）

このように、携帯電話会社各社のフォーマットにあわせた URL やパラメータによって、現在位置を取得し、得られた緯度と経度のデータを Web サーバへ送信する。

４．５　格納データ

格納データに関して、先行研究と同様に松江市を走行している一畑バス[7]と松江市営バス[8]の 2 つのバス会社のバス停や路線、時刻表のデータを格納している。テーブル名とテーブルの内容は以下の表 4.5 のとおりである。最近ではそれぞれのバス会社が平成 20 年 4 月 1 日にダイヤ改正を行っているので、今回のシステムではこのダイヤ改正に対応させている。

４．５．１　バス停位置テーブル

バス停位置テーブルは、各バス停の情報を格納した表である。バス停を検索する際にこの表を利用することになる。

このテーブルには、バス停 ID、バス停名、バス停の緯度・経度のデータを格納している。バス停 ID は各バス停毎にユニークに付けた ID を格納している。このユニークな ID は提供して頂いたデータに付けてあった ID とほぼ同じものを付けている。

次にバス停名は、提供して頂いたデータをそのまま使用している。ダイヤ改正に伴ってバス停名が変更されているところがある。

そして、バス停の緯度・経度に関しては、提供して頂いたデータは、日本測地系で計測したデータを 1/1000 秒単位の 10 進整数で記述してあった（例えば経度が 133 度 12 分 34。567 秒であったとすると度と分を秒に直し、1000 倍したもので、つまり(133×3600＋12×60＋34.567)×1000 を計算した値)。そのため、Google Maps で扱えるようにするためには世界測地系に変換する必要がある。

世界測地系への変換方法はまず､日本測地系で計測されたデータを 3600000 で割り、度で表す。次に、その座標の整数部分から 1 次メッシュコードを導く。さらに、その座標の整数部分をひいたものからそれぞれ緯度を 12 倍、経度を 8 倍し、その整数部分を 2 次メッシュコードとする。最後に、2 次メッシュコードを導いた座標から整数部分を引いたものからそれぞれ緯度を 120 倍、経度を 80 倍し、その整数部分を 3 次メッシュコードとする。

導いた 3 次メッシュコードでは地図からの検索をする上で、緯度と経度の 1 度あたりの長さが異なっているため、円形で検索をすると誤差が出てしまう。この問題に対して、3 次メッシュがほぼ 1km ということから、緯度・経度をそれぞれ 1200 倍、800 倍することで、1 が 100m となり、この値を格納している。

また、この緯度・経度は PostGIS を用いて検索ができるように GEOMETRY 型で定義し、格納している。このプラグインを使用し、比較的簡単な SQL 文を発行することで円形などの検索が可能となる。実際に格納した表は以下のようなもので、下表は松江市営バスの場合の表の一部である。一畑バスの表も同様に格納している。

表 4.5.1　バス停位置テーブル

バス停 ID （VARCHAR 型）	バス停名 （VARCHAR 型）	X 座標 （VARCHAR 型）	Y 座標 （VARCHAR 型）	座標 （GEOMETRY 型）
10012	上乃木	35.4469	133.0644	point(42536.3443 106451.5507)
:	:	:	:	:
11563	乃木公民館前	35.4451	133.0582	point(42534.1975 106446.5605)
:	:	:	:	:
14814	田和山史跡公園	35.4381	133.0526	point(42525.8004 106442.1304)

表 4.5.1 のようにユニークに付けたバス停 ID、バス停名は VARCHAR 型、表示に使用する座標である X 座標、Y 座標も VARCHAR 型、検索に使用する座標を GEOMETRY 型で定義し、格納した。主キーはバス停 ID とする。実際に格納したバス停数は、松江市営バスが 562 カ所、一畑バスが 684 カ所であった。

４．５．２　バス路線テーブル

バス路線テーブルは、時刻表を検索する場合に必要となる、バスの路線の情報を格納した表である。出発バス停と目的バス停を使った検索(両方のバス停を通る路線の検索)を行うためのデータを格納している。

具体的にはユニークに付けた路線 ID、路線名、バス会社、通過するバス停 ID の列で構成されており、通過するバス停 ID の列を検索することで、路線の検索を行っている。路線の検索において、通過するバス停 ID の列はスペースで区切った文字列として、格納している。SQL 文発行の際には、SQL 文の条件文で、”%出発バス停 ID%目的バス停 ID%”とすることで、検索がすることが出来る。ここで、%は 1 文字以上の文字列を表している。このデータは、提供頂いたデータを元に作成を行った。

実際に格納した表は以下のようなもので、松江市営バスの表の一部である。通過するバス停 ID のバス停数は各路線によって異なっている。また、一畑バスも同様に格納している。

表 4.5.2　バス路線テーブル

路線 ID （VARCHAR 型）	路線名 （VARCHAR 型）	通過するバス停 ID （VARCHAR 型）
10020101	県合同庁舎－川津　竪町～大橋	11612 11463 12552 10944 10952 10414 … 10270
:	:	:
13270101	竹矢－東高校　界橋・駅・くにびき	10822 10722 10882 11292 10932 11202 … 14700
:	:	:
16030101	レイクライン（嫁ヶ島）　松江駅止	11227 10374 14012 14022 14152 11452 … 11220

表 4.5.2 のようにユニークに付けた路線 ID、路線名、路線情報の列は全て VARCHAR 型で定義し、格納した。主キーは路線 ID とする。格納したバス路線数は、松江市営バスが 128 路線、一畑バスが 117 路線であった。

４．５．３　バス編成テーブル

バス編成テーブルは、各バス停の時刻表を格納している表である。バス停位置テーブルよりバス停 ID を検索し、その ID より、路線テーブルからどの路線の何番目に通るバス停 ID かを検索した後で、時刻表を検索する。

具体的には、路線 ID、時刻、バス ID、備考の列で構成されている。路線 ID はバス路線テーブルの路線 ID と同じもの(その時刻の路線名の ID)が格納されており、バス ID については同じ路線でも何本もバスがあるので何本目のバスなのかを決定するために用いる。時刻については通過するバス停と同形式で時刻が格納されている。そのため、何番目に通過するバス停かがわかれば、時刻を特定することが出来る。このデータに関しては、提供していただいたデータの状態を元に作成を行った。

実際に格納した表は以下のようなものであり、松江市営バスの表の一部である。時刻の数は各路線によって、異なっている。また、一畑バスも同じように格納している。

表 4.5.3　バス編成テーブル

路線 ID （VARCHAR 型）	備考 （VARCHAR 型）	バス ID （INTEGER 型）	時刻 1 （TIME 型）	…	時刻 132 （TIME 型）
10020101	平日	1	7:30:00	…	00:00:00
:	:	:	:	:	:
18370101	1、3、5 土曜	1	11:50:00	…	13:10:00
18370101	休日	1	11:50:00	…	13:10:00
18370101	平日	1	11:50:00	…	13:10:00
:	:	:	:	:	:
18740102	平日	2	16:30:00	…	00:00:00

表 4.5.3 のように、路線 ID、備考は VARCHAR 型で、バス ID は INTEGER 型、時刻は TIME 型で定義し、格納した。格納した時刻表の数は、松江市営バスが 911 個、一畑バスは 378 個であった。

４．５．４　バス路線位置テーブル

このテーブルは本研究で乗車部分の路線座標を抜き出す為に新たに作った表であり、バス路線のバス停全てを GEOMETRY 型の LINESTRING で定義している。先に検索された路線 ID と何番目のバス停かを元に検索を行う。

具体的には路線 ID、路線位置の列で構成されている。

実際に格納した表は以下のようなものであり、松江市営バスの表の一部である。格納したバス路線数は４．４．２のバス路線テーブルと同様である。また、一畑バスも同様に格納している。

表 4.5.4　バス路線位置テーブル

路線 ID （VARCHAR 型）	路線座標 （GEOMETRY 型）
10020101	LINESTRING(35.4477 133.0852,35.4453 133.0846,……,35.4854 133.0709)
:	:
13270101	LINESTRING(35.4387 133.1203,35.4409 133.1185,……,35.4842 133.0783)
:	:
16030101	LINESTRING(35.4642 133.0624,35.4699 133.0652,……,35.4641 133.0631)

表 4.5.4 のように、路線 ID は VARCHAR 型、路線位置は GEOMETRY 型、LINESTRING で定義し、格納した。

これらの表をまとめると、以下の表 4.5 のようになる。

表 4.5　テーブル一覧

テーブル名	内容
バス停位置	バス停の座標を格納した表。
バス路線	バス路線の停車バス停を格納した表。
バス編成	バス停の時刻表を格納した表。
バス路線位置	バス路線の路線座標を格納した表。

※なお、一畑バスの場合、テーブル名の後に数字の「11」がつき、松江市営バスの場合はテーブル名の後に数字の「13」がつく。また、今回はダイヤ改正に対応させたテーブル名を一畑バスの場合は「テーブル名 11_20」、松江市営バスの場合も同様に「テーブル名 13_20」とした。

４．６　検索方法

① 出発地と目的地で共に半径 500m 以内にバス停があるか判定する。なければ終了。
出力されるメッセージは「半径 500m 以内にバス停がありませんでした。」となる。

② 受け取った出発位置からまず、半径 100m 以内の周辺のバス停を検索する。なければ、半径 200m 以内の周辺のバス停を検索する。これを 100m 毎に 500m まで行い、１件も見つからなければ、終了する。
出力されるメッセージは「半径 500m 以内にバス停がありませんでした。」となる。

③ 目的位置からも出発位置と同様にまず、半径 100m 以内の周辺のバス停を検索する。なければ、半径 200m 以内の周辺のバス停を検索する。これを 100m 毎に 500m まで行い、１件も見つからなければ、終了する。
出力されるメッセージは「半径 500m 以内にバス停がありませんでした。」となる。

④ 周辺のバス停が見つかった場合、そのバス停と出発のバス停との路線を検索する。路線があれば、時刻表を検索し、なければ、次の目的位置周辺のバス停を検索する。それでもない場合は②へ戻る

⑤ 路線があった場合、この路線の何番目のバス停かを取得する。路線がなかった場合は「路線がありませんでした。」と表示される。

⑥ 何番目のバス停かと日時を元に時刻を検索する。

⑦ 路線 ID と何番目のバス停かを元に路線座標を検索する。

このような手順によって検索を行う。図 4.6 にフローチャートを示す。

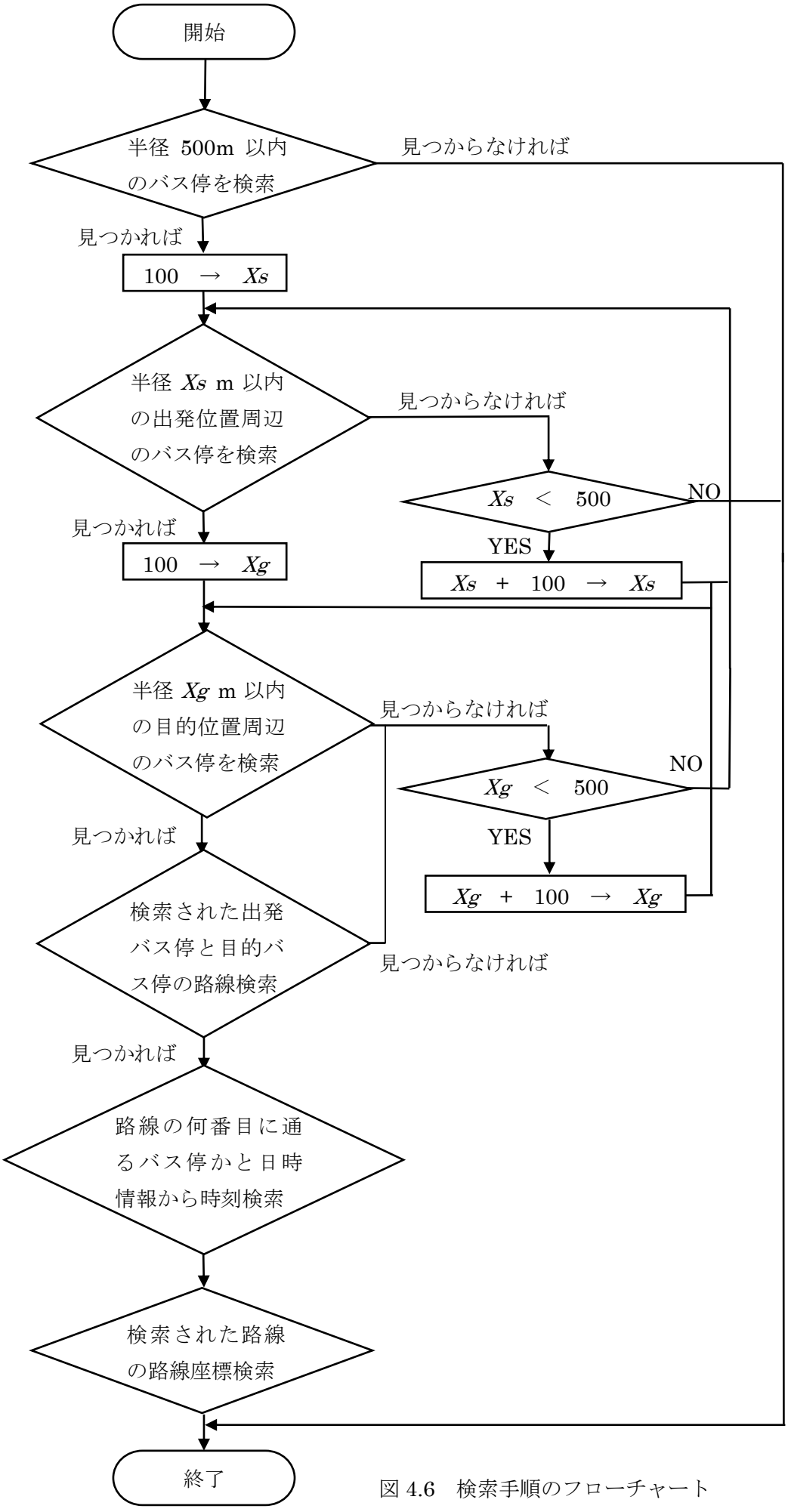


図 4.6　検索手順のフローチャート

４．６．１　周辺のバス停位置検索

周辺のバス停位置の検索には、取得した位置情報と各バス停の位置情報を PostgreSQL の空間情報の拡張である PostGIS を利用して DBMS で検索を行っている。まず、取得した位置情報を変換して、各バス停の位置情報を用いた検索を出来るようにする。ここでは例として半径 500m 以内のバス停検索を行う。

実際に PostgreSQL へ発行する SQL 文は以下のようなものであり、図で表すと図 4.6.1 のようになる。

```
SELECT バス停名 FROM バス停位置 WHERE ST_DWITHIN(座標,
GeomFromText('POINT(緯度 経度)',SRID[4326]),半径[500m]) = TRUE
```

※”バス停名”は列名、”バス停位置”は表名、”WHERE”以下が検索の条件である。また、”GeomFromText( )”は PostGIS 独自のもので空間検索する場合に( )内に Geometry、SRID、半径などの条件を指定する。”POINT”も PostGIS 独自のものであり、円形を表す。この( )内は緯度・経度を指定する。

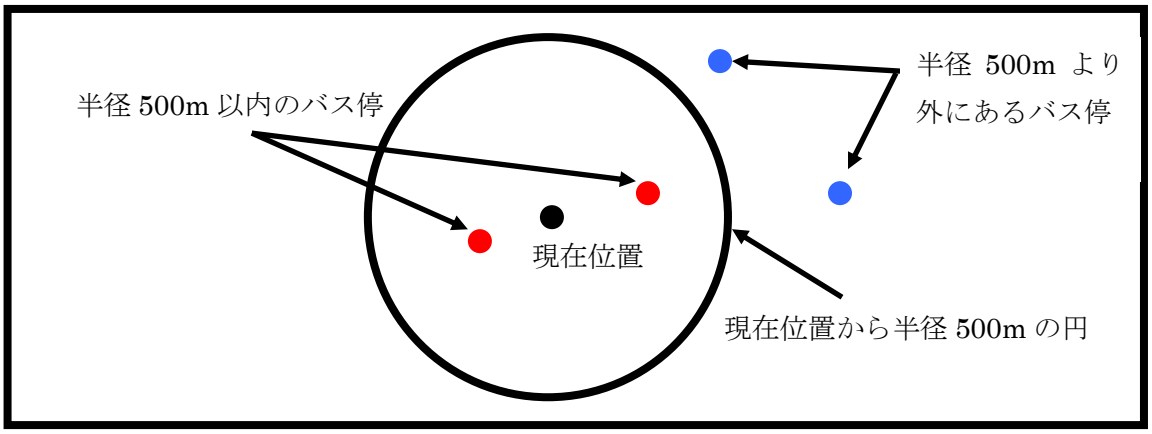


図 4.6.1　空間検索

４．６．２　路線座標検索

路線座標の検索には、バス路線テーブルから取得した路線 ID と何番目に通るバス停かを元に出発バス停から目的バス停までの路線座標を検索する。

実際に PostgreSQL へ発行する SQL 文は以下のようなものであり、検索方法は図 4.6.2 のとおりである。

```
SELECT AsText(路線座標) FROM バス路線位置 WHERE 路線 ID = 10020101
```

※”路線座標”は列名、”バス路線位置”は表名、”WHERE”以下が検索の条件である。また、AsText( )はカプセル化された geometry 型データを WKT 文字列で返す。

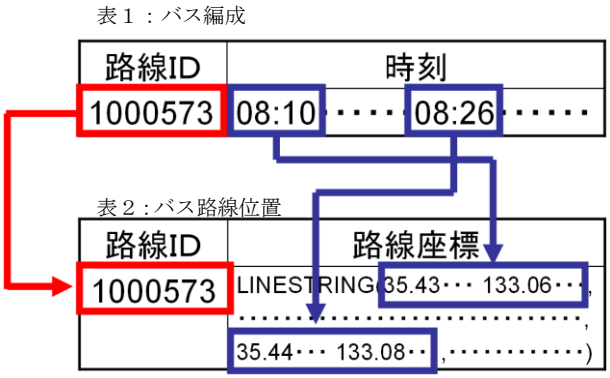


図 4.6.2　路線座標検索方法

４．７　結果表示

PC での結果表示には JavaScript や Google Maps API などを用いている。携帯端末での結果表示には Google Maps を画像化したものを用いている。

４．７．１　PC の場合

PC では Google Maps API を用いて表示を行っている。なお、時刻の表示には Ajax による非同期通信を利用している。非同期通信とはクライアントとサーバ間で同期を取らずに通信することである。

同期的な通信の場合、ブラウザに表示しているページから他のページへ移動する際にリロードする必要があるのに対し、非同期通信の場合は、他のページへ移動する際にも絶えずサーバとの通信を行うため、ページ遷移が発生しない。

同期通信と非同期通信の違いを図で表すと図 4.7.1 のようになる。また、PC での結果表示を図 4.7.2 に示す。

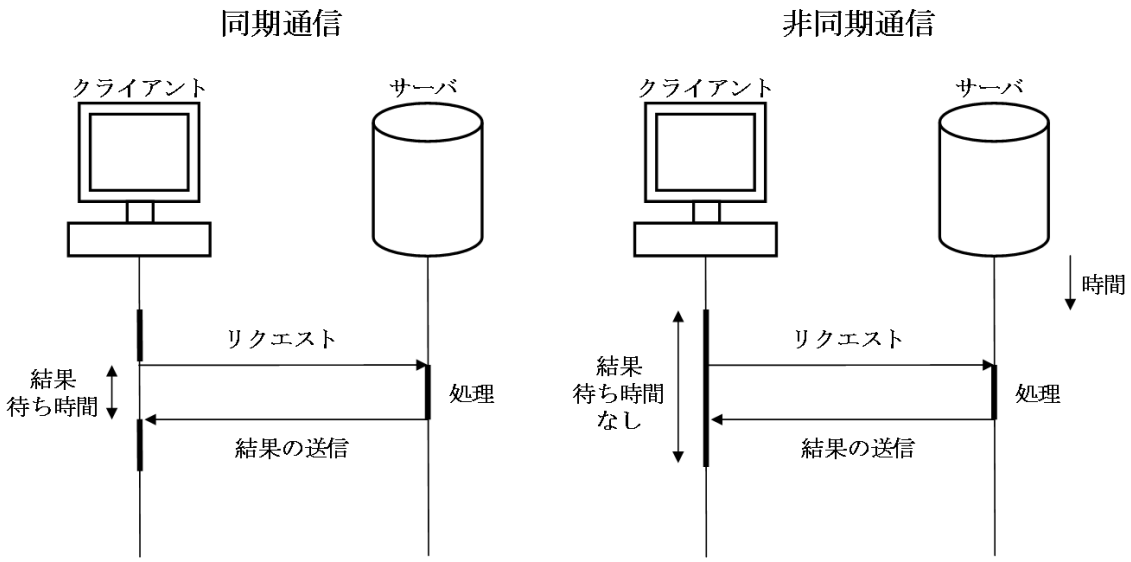


図 4.7.1　同期通信と非同期通信の違い

図 4.7.2　PC での結果表示

４．７．２　携帯端末の場合

携帯端末では、指定した座標の Google Maps を画像化し、送信するという機能を用いて携帯端末に地図画像を表示している。この機能の中に地図上にアイコンを表示したりする機能もついている。

なお、Google Maps の画像化には Google Static Maps API を用いている。この API によって、Google Maps の画像を自分の Web ページに埋め込むことができる。この API では JavaScript や動的なページの読み込みは必要ない。

また、Google Static Maps API の URL の記述方法は以下のとおりである。

http://maps.google.com/staticmap?*parameters*

parameters の部分については、

・center：地図の中心(複数のマーカーが存在しない場合は必須)

・zoom：地図の拡大レベル(複数のマーカーが存在しない場合は必須)

・size：地図画像の矩形寸法(必須)

・format：画像の形式(省略可)
・maptype：地図のタイプ(省略可)
・markers：マーカーの定義(省略可)
・path：経路の定義(省略可)
・span：緯度と経度の組み合わせの地図画像の最小の「ビューポート」の定義(省略可)
・frame：画像の枠の指定(省略可)
・hl：マップタイルのラベルを表示する言語の定義(省略可)
・key：Google Maps API キーの指定(必須)
・sensor：位置情報を取得するのにセンサーを用いているかの指定(必須)

これらのパラメータによって以下の図 4.7.3 のように Google Maps を画像化して表示することが出来る。

図 4.7.3　携帯端末での結果表示

第５章　データ更新

松江市内で路線バスを運行している一畑バス、松江市営バス共に平成 20 年 4 月 1 日にダイヤ改正を行っており、先行研究ではこのダイヤ改正以前のデータを使用している為、データ更新を行った。なお、追加のバス停の一部に座標が不明なバス停があった為、MAPPLE[9]を参考にしてバス停座標の格納を行った。

更新前と更新後のデータ数の違いは表 5.1 のとおりである。

表 5.1　データ数比較表

テーブル名	一畑バス		松江市営バス	
	更新前	更新後	更新前	更新後
バス停位置	622	684	476	562
バス路線	126	117	102	128
バス路線位置	126	117	102	128
バス編成	818	378	1238	911

バス停の廃止や名称変更、バス路線の統廃合や時刻の改正などにより、データ数が変わっている。

第６章　今後の課題

本研究では、沿線情報検索システム作成の準備として、先行研究のシステムに路線座標を検索して表示する機能を追加した。沿線情報検索システムの作成が今後の課題になるので、最後に沿線情報を検索する方法を提案する。

６．１　沿線情報検索方法の提案

６．１．１　Buffer と Within 関数を用いる方法

この方法では、路線の LINESTRING データに Buffer を発生させ、Polygon データに変換する。次に Within 関数を使って Polygon 部分を検索する。

Within 関数のシンタックスは以下の通りである。

Within(geometry A,geometry B)

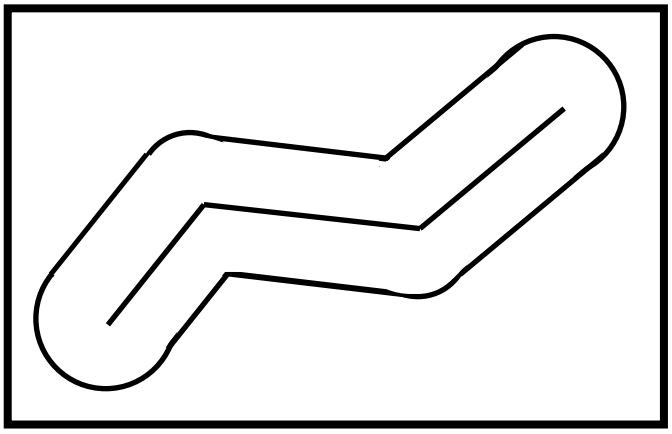
・geometry A：geometry 型データ

・geometry B：geometry 型データ

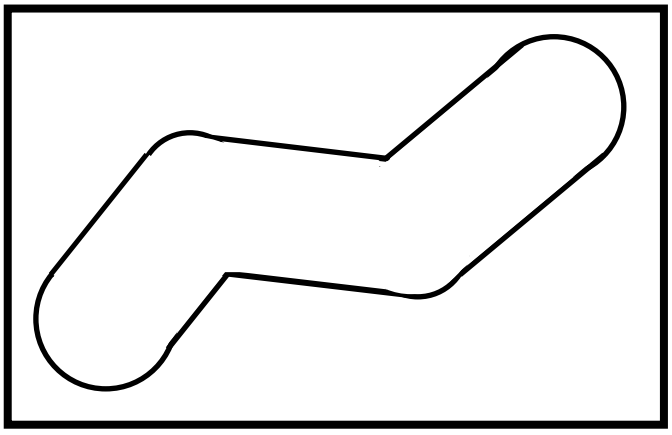
意味：geometry B の中に geometry A が完全に含まれれば TRUE を返す。

また、図に表すと以下のようになる。

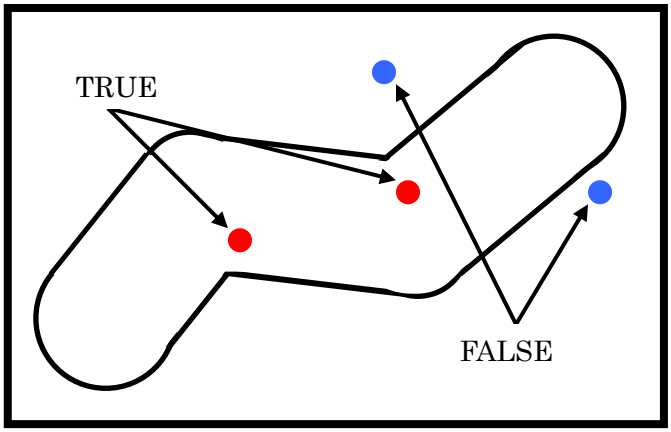
手順 1：LINESTRING データに Buffer を発生させる。



手順 2：Polygon データに変換する。



手順 3：Within 関数を使って Polygon 部分を検索し、TRUE or FALSE を返す。



６．１．２　DWithin 関数を用いる方法

この方法では、DWithin 関数を使って GEOMETRY 型データ同士の距離を指定して検索する。この方法は６．１．１の方法の Polygon データに変換するというステップがないので６．１．１の方法より時間が短縮できると考えられえる。

DWithin 関数のシンタックスは以下の通りである。

DWithin(geometry A,geometry B,distance)

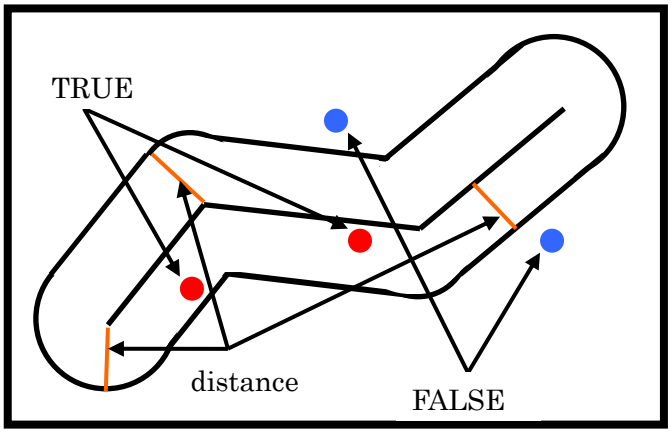
・geometry A：geometry 型データ

・geometry B：geometry 型データ

意味：geometry B の距離 distance m 以内に geometry A があれば TRUE を返す。

また、図に表すと以下のようになる。

手順 1：DWithin 関数を使って GEOMETRY 型データ同士(LINESTRING と POINT)の距離を指定して検索し、TRUE or FALSE を返す。



６．２　POI 情報の取得

POI とは Point of Interest の略であり、観光地、商店などのスポット（点）情報である。車のナビゲーションシステムではこの POI 情報をネットから取得して表示することができる。POI 情報には施設情報や位置情報のデータが格納されているので、沿線情報検索システムで、この POI 情報を取得し、検索できると有益である。

第７章　終論

本研究では先行研究を今まで用いていた有償の HiRDB ではなく、無償でオープンソースの PostgreSQL で再現することにより、HiRDB と同様の機能を PostgreSQL でも実装できることが分かった。これは企業などでのコスト削減にもつながると思う。

また、第６章でも述べたように、本研究では、沿線情報検索システム作成の準備として、先行研究のシステムに路線座標を検索して表示するシステムを作成することが出来た。このシステムと第６章であげた手法を用いて沿線情報検索システムの作成をすることが今後の課題である。作成したシステムは PC と携帯端末の両方に対応している為、ユーザーは家にいる時でも外出している時でも場所にとらわれず利用することが可能である。今後、現在の携帯端末より高機能な端末も開発されると考えられるので、それらの機能の長所を生かしたシステムを作ることも重要であると思う。また、沿線情報を検索する際に第６章の６．２で述べたように POI 情報を取得し、検索できると有益である。また、バス路線だけでなく、列車や他の交通機関に対応させ、松江市だけでなく他の県や市町村にエリアを拡大するとより実用的になると思う。

また、今回データの更新を行ったが、かなりの時間を要した。電車やバスのダイヤ改正は定期的に行われるものなので、運行会社側が DB にアクセスして、簡単にデータ更新を行えるようなツールがあると有益になると思う。

また、本システムでの路線座標の表示はバス停同士を直線で結んでいる為、道路に沿っていない部分が多々ある。道路の座標を格納し、表示させたほうがより見やすくなるので改善する必要があると思う。

第８章　謝辞

本研究にあたり、最後まで熱心な御指導をいただきました田中章司郎教授には心より御礼申し上げます。また、同じ研究室の李昊さん、下田圭亮さん、両間成利さん、韋于思さんには本研究に関して、数々の御協力と御助言を頂きました。厚く御礼申し上げます。

なお、本論文、本研究で作成したプログラム及びデータ、並びに関連する発表資料などの全ての知的財産権を本研究の指導教官である田中章司郎教授に譲渡致します。

参考文献・資料

[1] 木村 眞吾、「目的別空間データベースサーバを用いた Web システムの設計と実装」島根大学　総合理工学部 数理・情報システム学科 卒業論文 (2007 年)

[2]「Google Maps」
http://maps.google.co.jp/

[3]「Apache」
http://www.apache.org/

[4]「Apache Tomcat」
http://tomcat.apache.org/

[5]「PostgreSQL」
http://www.postgresql.org/

[6]「PostGIS」
http://postgis.refractions.net/

[7]「一畑バス株式会社」
http://www.ichibata.co.jp/bus/

[8]「松江市交通局」
http://www.matsue-bus.jp/

[9] MAPPLE 地図
http://www.chizumaru.com/index.aspx

[10] GPS 携帯 位置情報 基礎知識
http://www.yaskey.cside.tv/mapserver/note/gps.html

[11]au　簡易位置情報
http://www.au.kddi.com/ezfactory/tec/spec/eznavi.html

[12]SoftBank　位置情報
http://creation.mb.softbank.jp/web/web_position.html

[13]docomo　位置情報
http://www.nttdocomo.co.jp/service/imode/make/content/browser/html/tag/location_info.html