変更履歴

t\_station.same\_flgはintからcharに変更空文字列が重複駅なし。文字列が含まれていると重複駅ありとなる。

文字列は駅名表示時、他の駅と区別のつかないように記した文字列（「(東)」とか、「(海)」とか）

この変更の為に、駅名表示時、クエリ結果列を、ltrim(name)+same\_flgとすることができる

jctflgはcityflg, spe\_routeへ移す

ダイクストラの高速化のため

また会社線乗り換えを考慮すること

新横浜から京都までの運賃計算算出方法は？

次の経路は、片道乗車券として、○発券できる、×発券できない。

○岐阜→名古屋→（新幹線）→米原→岐阜

×名古屋→（新幹線）→米原→（在来線）→名古屋

* 米原‐名古屋間（両駅除く）の在来緯線駅発着または合流・分岐の場合のみ許される

# UI(iPhone)

151条（分岐駅）

経路指定時、自動計算（経路は変更しないが適用情報を保持して開示可能にする）

70条

経路指定時、自動計算（経路は変更しないが適用情報を保持して開示可能にする）

新幹線と在来線乗換

経路指定時、自動計算（経路は変更しないが適用情報を保持して開示可能にする）

69条

経路指定時、自動計算（経路は変更しないが適用情報を保持して開示可能にする）

157条2項

86条／87条

経路指定時は適用しない。以下ボタンをクリックすることにより適用／非適用が切り替わる

「北九州市内発」／「単駅発」、と「北九州市内着」／「単駅着」の2つのボタンが有効化される

経路指定時はすべて指定経路の営業キロと運賃を表示

69条、70条は自動適用して計算

86条、87条も自動適用して計算（但し、非適用ボタンがある）

単駅指定 ボタンを有効化

都区市内発 ボタン有効化

都区市内着 ボタン有効化

114条 ボタンを有効化

157条 ボタンを有効化

最短 ボタンを有効化

同名駅は漢字上での話で読みが異なっても同名とはしない

C++ライブラリ、

SQLITE3 C++ Wrapper class を使用（The Code Project 版のものを使用：改変自由、商用使用可）

発駅 TextField　選択ボタン>>

経由

運賃・有効期間表示エリア

# データ数、制限値

厳密には正確ではないが設計には有用な値

|  |  |  |
| --- | --- | --- |
| **項目** | **数** |  |
| 路線数 | 202 |  |
| 駅数 | 4589 |  |
| 分岐駅数 | 310 |  |
| t\_lines | 4965 | (旧) 新幹線、分岐駅数分＋ |

制限値

|  |  |
| --- | --- |
| 項目 | 数 |
| 路線・駅数（t\_linesレコード数） | 16383=14ビット(0x3fff) |

* 制限値を越えるようだと設計の見直しが要となる。

# データ構造

## データ関連図

RouteItem\*

View

Route1

図 1

RouteItem

通常ルート(ルートtypeL or typeI)

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0x0000 | **長津田** | 発駅 |
| 1 | 横浜線 | 東神奈川 |  |
| 2 | 東海道 | 川崎 |  |
| 3 | 南武 | 立川 |  |
| 4 | 中央東 | **高尾** | 着駅 |

閉じたルート(ルートtypeO or typeP)

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0xffff | **長津田** | 発駅 |
| 1 | 横浜線 | 東神奈川 |  |
| 2 | 東海道 | 川崎 |  |
| 3 | 南武 | 立川 |  |
| 4 | 中央東 | 八王子 |  |
| 5 | 横浜線 | **長津田** | 着駅 |

## QueryResultSetオブジェクト

説明

SQLiteクエリー結果を得るイテレータオブジェクト

クラス変数

string qryStationsFromLineAndPrefect

string qryStationsFromStation

string qryStationFromLine

string qryLineFromPrefect

string qryLineFromCompany

これらは、QueryResultSetオブジェクトをアプリケーションレイヤから使用する場合に利用

内部で使用する場合、直接SQLステートメント文字列を指定する

インスタンス変数

- string sql\_statement;

コンストラクタ

QueryResultSet(const string& sql);

QueryResultSet();

メソッド

vector<string>\* exec(const string& sql, const vector<string>& param);

vector<string>\* exec(const vector<string>& param);

結果は、配列の配列(2次元配列)で返します。エラーの場合、nullを返します。

## RouteItemオブジェクト

説明

RouteItemオブジェクトはルート（経路）を定義するオブジェクトで、始発駅から着駅まで動的配列またはリスト）で保持する。

クラス変数

なし

インスタンス変数

int16 stationId; //発着駅(1つめは発駅)

int16 lineId; // 路線(1つめは以下※)

// 0の場合、営業キロの算出はおこなわない（旅客営業規則70条処理）

int16 optStationId; //「新幹線分岐特例」時の表示上分岐駅（例：恵那→三島時、lineId=「新幹線」,stationId=「金山」時の「名古屋」）

int16 flag; //spe\_routeとcityflag,jctflagなどを集約(使用するのもの)

|  |
| --- |
| bit0-15: cityflag.b0-15  都区市内着発適用時は都区市内番号が設定される  bit24-31: spe\_route.b31-24 |

※ 始発駅のlineIdには、以下の値が格納される

ビット0： 発駅が都区市内

ビット1: 着駅が都区市内

ビット2: 発駅が山手線内

ビット3: 着駅が山手線内

ビット4: 都区市内通過（発駅）

ビット5: 都区市内通過（着駅）

0: I字ルート

-1: これ以上先へはいけないルート（O字、P字）

int16 alternate\_stationId; //実際の分岐駅（例：「名古屋（金山）」の名古屋）、0の場合は通常

メソッド

void stationName(string\* station, string\* station\_line); // 駅名と駅の所属する’/’で区切った複数の路線名を返す。

void lineName(string\* line); // 路線名を返す。発駅の場合は空文字を返す。

## Routeオブジェクト

クラス変数

なし

インスタンス変数

‐ int passingJunction\_[(MAX\_JCT + 7) / 8]; // ルートリスト

- vector<RouteItem> route\_items;

- int16 y\_branch\_station; // b: calc\_km(分岐駅)(新旭川)

（最も最近の特例分岐駅）

- int16 y\_branch\_line; // c: sales\_km（本線）(宗谷)

（最も最近の特例分岐駅）

* 最終追加route\_itemsのspe\_route.BSRJCTSPがOFFのときは、y\_branch\_station、y\_branch\_lineは、0に初期化する(Mustではないはず)

メソッド

void setStartStation(int staionId) // 始発駅の追加

void clear(); //ルートリストを消去

int add(int lineId, int stationId1, int stationId2); // ルートの追加

void removeTail(); // ルートの最終項目を削除

int itemAt(int index, const RouteItem& route\_items); // RouteItemを得る

int size(); // ルート数を得る

void calcFare(FARE\_RESULT\* result); // 運賃計算

イテレータメソッド

RouteIterator\* iterator(); 開始のイテレータを返す。RouteIteratorオブジェクトはシングルトンとして静的ポインタを返す（delete不要）

## RouteIteratorオブジェクト

説明

構築時にイテレータインデックスを初期化する

クラス変数

なし

インスタンス変数

‐ int current\_index;

- vector<RouteItem>\* route\_items;

メソッド

bool hasNext(); //つづきの有無を返す

RouteItem\* next(); //カレントRouteItemを返す

使用方法：

RouteIterator \*routeIterator = route.iterator();

RouteItem\* routeItem;

while (routeItem = routeIterator->next()) {

routeItem->xxx

:

}

## void Route::setStartStation(int staionId) 始発駅の追加

ルートリストの開始駅を設定する。既に設定されている場合は上書きし、途中まで格納されていたルートリストがあればそれもすべて削除される。

## ルートリスト（int passing\_junction\_[(MAX\_JCT + 7)/8]）

発駅と着駅を除き分岐駅のみに通過マークをセットする。分岐マークは40バイトの配列にビット単位で設定する。

分岐駅はt\_jctテーブルにstationIdとIdとして変換できるようにしている。

分岐駅の総数は311で、MAX\_JCTには320と定義される。(MAX\_JCT + 7) / 8=40

## 乗車経路マーク

Int Route.add(int lineId, int stationId1, int stationId2)

乗車経路をマークする。のちの片道乗車券判定時の重複経路になっていないかの判定に使用する。片道乗車券とは、一度通った駅を通らない一筆書きの経路である。例として、Lの字、Iの字、Oの字は許されるが、Qの字や、9の字は許されない。但し6の字は許される。

路線の分岐駅 && 営業キロ ≦ 駅1 &&　駅2 ≦ 営業キロ order by sales\_km ||

路線の分岐駅 && 営業キロ ≦ 駅2 &&　駅1 ≦ 営業キロ order by salse\_km desc

で列挙される駅の分岐IDを足跡配列の該当ビットオフセットをONにする。

既にONになっていたら重複エラーとなる。正し、発駅がONの場合は良しとし、着駅がONの場合は最後とする（最終駅警告）（ルート終了。閉じたルート。いわゆるルートtypeOまたは、typeP）

足跡配列数は、(MAX\_JCT + 7)/ 8

MAX\_JCTは、311なので、余裕を見て320とってある。320ビット=40バイトの配列

クエリー文を以下にしめす

パラメータは、?1=lineId, ?2=station1, ?3=station2

select id, t.name from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_jct j on j.station\_id=l.station\_id where

line\_id=?1 and jctflg!=0 and

case when

(select sales\_km from t\_lines where line\_id=?1 and station\_id=?3) <

(select sales\_km from t\_lines where line\_id=?1 and station\_id=?2) then

(((sales\_km>=(select sales\_km from t\_lines where line\_id=?1 and station\_id=?3)) and

(sales\_km<=(select sales\_km from t\_lines where line\_id=?1 and station\_id=?2))) ) else

(((sales\_km<=(select sales\_km from t\_lines where line\_id=?1 and station\_id=?3)) and

(sales\_km>=(select sales\_km from t\_lines where line\_id=?1 and station\_id=?2))) ) end

order by

case when

(select sales\_km from t\_lines where line\_id=?1 and station\_id=?3) <

(select sales\_km from t\_lines where line\_id=?1 and station\_id=?2) then

sales\_km

end asc,

case when

(select sales\_km from t\_lines where line\_id=?1 and station\_id=?3) >

(select sales\_km from t\_lines where line\_id=?1 and station\_id=?2) then

sales\_km

end desc;

* 以前は上り用と下り用の二つのクエリを発行していたが一回でOKのようにした（ただし同じ条件文が複数回あり、冗長なので、プリコンパイル済みクエリーが使用するべき。

初期化・クエリー発行

c = 0

lineIdのstation1～station2間の分岐駅を列挙: ite

ite ≠station1(1つめ以外)

station2は通過済みではない

c++

lineIdのstation1～station2間の分岐駅を列挙

通過マークOFF

c0++

c0 < (c – 1)

c0 = 0

c ≦ c0

return -1

(経路重複エラー)

通過マークON

通過済みはstation2(最後)ではない

分岐駅≠station1(1つめ以外)

通過済みはstation2(最後)

Routeオブジェクトの先頭RouteItemのlineId=0xffffに設定（ルート終了、閉じたルート）

int Route::add(lineId,station1, station2)

lineDupCheck(lineId, stationId1, stationId2)

Success

エラー

着駅(station2)エラー

Routeオブジェクトの先頭RouteItemのlineId=0xffffに設定(ルート終了、閉じたルート)

RouteオブジェクトへRouteItem(lineId, stationId2)追加

return 0（Success）

RouteItem配列は空

RouteオブジェクトへRouteItem(lineId, stationId1)追加

station2が通過済みでは場合、lineIdは0x0000とする．

return 1（station2は通過済み）(ルート終了、閉じたルート)

※ルート終了、閉じたルート­＝Route typeO、typeP

通過済

図 2

* **フローは細かすぎてコーディングできないのでみなおし**

関数値

0： 成功

‐1： 経路重複

1： 経路重複（stationId2, typeO, P）

MSB(if 0 or 1)=1(0x8000\_0000 or 0x8000\_0001) ： 特例によるルート変更あり。→ 表示時、全アイテムをリフレッシュして一覧更新する必要がある。それ以外は、最終アイテムの追加のみの対応で可能

**Int　Route::lineDupCheck(lineId, stationId1, stationId2)**

長津田-**東神奈川**-**川崎**-**立川**-**八王子**-十日市場みたいなの（八王子-十日市場はNG）を検知する。

十日市場-東神奈川間はNG

八王子-長津田間はOK

橋本の次-長津田間は閉塞区間なので終着

駅1が最初の路線と同一路線に限り、最初の発駅〜降車駅、駅1、駅2を営業キロでソートし、駅1、駅2が発駅〜降車駅内にないこと(2本の線が重なり合わないこと)。

→新幹線、在来線の考慮も必要になる例は？

発駅に限ってチェックしているのは、途中分岐駅は分岐駅チェックでひっかかるので分岐駅以外というのは発駅と着駅以外にないので。

lineDupCheck(lineId, staion1, staion2)

Route配列サイズ＜2

最初の乗車路線==lineId?

配列[0] firstTakeStation(発駅=最初の乗車駅)

配列[1] firstLeaveStation(最初の降車駅)

firstTakeLine(最初の乗車路線)

firstTakeStationまたはstation2は分岐駅?

最初の発駅、最初の降車駅、station1, station2を営業キロでソートクエリーして4の配列へ格納

駅数?

=3

3未満(あり得ない)

return -1(Error)

=4

配列[0]はfirstTakeStation?

N

N

Y

return 0(Success)

Y

発駅=station2 ?

typeO ?

return -1(Error)

return 1(着駅は重複)(ルート終了、閉じたルート）

Y

N

N

return 0(Success)

N

Y

配列[1]はfirstTakeStation?

配列[2]はfirstTakeStation?

配列[3]はfirstTakeStation?

配列[2]はfirstLeaveStation?

配列[3]はfirstLeaveStation?

配列[1]はfirstLeaveStation?

配列[0]はfirstLeaveStation?

N

return 0(Success)

return 0(Success)

return -1(Error)

N

N

N

N

Y

Y

Y

Y

Y

最初の降車駅=station2

(あり得ない)

N

Y

Y

図 3

上記フローは不採用

以下にクエリーを高等化した簡易処理版をしめす。

以下のクエリーで、横浜線（?1）長津田(?2)～東神奈川(?3)間に駅(?4)が含まれていないかをチェックします。

select count(\*) from (

select station\_id

from t\_lines

where line\_id=?1

and station\_id

in (select station\_id

from t\_lines

where line\_id=?1

and ((sales\_km>=(select sales\_km

from t\_lines

where line\_id=?1

and station\_id=?2)

and (sales\_km<=(select sales\_km

from t\_lines

where line\_id=?1

and station\_id=?3)))

or (sales\_km<=(select sales\_km

from t\_lines

where line\_id=?1

and station\_id=?2)

and (sales\_km>=(select sales\_km

from t\_lines

where line\_id=?1

and station\_id=?3)))))

)where station\_id=?4;

含まれていれば、1を返し、含まれていなければ0を返す。

含まれている駅(?4)を増やすには、クエリ文に 「or station\_id=?5」を追加するのみである

lineDupCheck(lineId, staion1, staion2)

Route配列サイズ＜2

firstTakeLine==lineId?

配列[0] firstTakeStation(発駅=最初の乗車駅)

配列[1] firstLeaveStation(最初の降車駅)

firstTakeLine(最初の乗車路線)

firstTakeStationまたはstation2は分岐駅?

lineId(?1), firstTakeStation(?2), firstLeaveStation(?3), station2(?4)でクエリ

0?

return -1(Error)

=1

return 0(Success)

return 1(着駅は重複)(ルート終了、閉じたルート）

N

return 0(Success)

N

Y

N

Y

Y

firstTakeStation==station2 ?

Y

Y

N

N

**void Route::removeTail();**

ルート配列の末尾を削除する。

RouteItem配列数 ＜ 2

lineIdのs1～s2間の分岐駅を列挙: ite

void Route::removeTail(void)

RouteItem配列数=1 or ((ite≠s1) and (ite≠s2 or RouteItem配列[0]のlineId=0xffff(typeO or typeP)))

removeAll() to RouteItem配列

RouteItem配列[-1]のstationId1 → s2

RouteItem配列[-2]のstationId2 → s1、lineId → lineId

通過マークOff

RouteItem配列[-1]を削除(最後のItemを削除)

図 4

#### 分岐特例駅の取得

**a,dから、b, cの取得（t\_linesテーブルより）**

ex.) spe\_route.BSRJCTSP=ONの“石北線-旭川”のレコードの実分岐駅”c新旭川”、通過列車分岐駅の所属路線”bを得る

<入力>

a:分岐路線（石北）

d:乗換駅（旭川）

<出力>

b: sales\_km（本線）(宗谷)→ route.y\_branch\_lineへ保持

c: calc\_km(分岐駅)(新旭川)→ route.y\_branch\_stationへ保持

* spe\_route.BSRJCTSPがOFFのときは、y\_branch\_station、y\_branch\_lineは、0に初期化される(Mustではないはず)

#### 路線ｌのa駅～b駅間に駅cはあるか？

int isInStation(int lineId, int station1, int station2)

return 0=no / 1=あり

* 「lineDupCheck()」で使用したクエリーをつかう

#### 路線のa駅→b駅は、上りか下りか？

int dirLine(int lineId, int station1, int station2)

return 0=station1=station2 / 1=上り/-1=下り

以下のクエリーは、路線?1の駅?2→駅?3への乗車が上りなら1を下りなら0を返す。

（駅1=駅2の場合、下り(0)を返す）

select case when

((select sales\_km from t\_lines where line\_id=?1 and station\_id=?2) -

(select sales\_km from t\_lines where line\_id=?1 and station\_id=?3)) < 0

then 0 else 1 end;

#### 新幹線の判定

line\_idのある値以上を新幹線とする。ある値は、~~起動初期化時、t\_lineテーブルをid順に列挙し、最初に「新幹線」を含むレコードのidを保持することにより決定される。~~#define(final int)定義する。

データ作成時、新幹線はt\_lineの後半になるようにする。

将来、新規路線が追加になった場合は、すべてのデータを再構築する。

# ■重複路線

路線名称と区間が重複する場合の対応

新幹線区間のダイクストラ検索時など

例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 東京 | → |  | 東京 | → |  | 東京 |
| 東海道 | 熱海 | 東海道 | 熱海 | 東海道 | 名古屋 |
| 新幹線 | 静岡 | 東海道 | 静岡 |  |  |
| 東海道 | 掛川 | 東海道 | 掛川 |
| 新幹線 | 名古屋 | 東海道 | 名古屋 |

新幹線、熱海－静岡は、東海道線に置き換えるべき

では、

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 新横浜 | → |  | 新横浜 |
| 新幹線 | 新富士(東) | 新幹線 | 小田原 |
|  |  | 東海道 | 三島 |
|  |  |  | 新幹線 | 新富士（東） |

新幹線、新横浜‐熱海は、新横浜‐小田原、東海道線‐

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 東京 | → |  | 東京 |
| 長野北陸新幹線 | 長野 | 東北線 | 大宮 |
|  |  | 高崎線 | 高崎 |
|  |  |  | 長野北陸新幹線 | 長野 |

新幹線を指定しているのであれば、新幹線のままの表記の方が誤解することがない。

知っているユーザであれば、路線を指定するはずである。

**したがって指定したまんまとする。**

ダイクストラで新幹線と入り混じる件についてはアルゴリズムの優先度で対処できる問題であり、これは新幹線優先とする。

# ■基定151条（分岐駅を通過する列車に乗車する場合の特例）対応

### **分岐駅を通過する列車に乗車する場合の特例区間一覧**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **a分岐路線** | **c分岐駅** | **d乗換駅** | **b本線** |
|  | 釧網線 | 東釧路 | 釧路 | 根室線 |
|  | 石北線 | 新旭川 | 旭川 | 宗谷線 |
|  | 千歳線 | 白石 | 札幌 | 函館線 |
|  | 札沼線 | 桑園 | 札幌 | 函館線 |
|  | 千歳線 | 沼ノ端 | 苫小牧 | 室蘭線 |
|  | 江差線 | 五稜郭 | 函館 | 函館線 |
|  | 海峡線 | 中小国 | 蟹田 | 津軽線 |
|  | 五能線 | 川部 | 弘前 | 奧羽線 |
|  | 男鹿線 | 追分 | 秋田 | 奧羽線 |
|  | 仙山線 | 羽前千歳 | 山形 | 奧羽線 |
|  | 左沢線 | 北山形 | 山形 | 奧羽線 |
|  | 水郡線 | 安積永盛 | 郡山 | 東北線 |
|  | 陸羽西線 | 余目 | 酒田 | 羽越線 |
|  | 上越線 | 宮内 | 長岡 | 信越線@ |
|  | 烏山線 | 宝積寺 | 宇都宮 | 東北線@ |
|  | 両毛線 | 新前橋 | 高崎 | 上越線@ |
|  | 八高線 | 倉賀野 | 高崎 | 高崎線@ |
|  | 横浜線 | 東神奈川 | 横浜 | 東海道線 |
|  | 中央東線 | 神田 | 東京 | 東北線\* |
|  | 中央東線 | 代々木 | 新宿 | 山手線\* |
|  | 中央西線 | 塩尻 | 松本 | 中央東線 |
|  | 中央西線 | 金山 | 名古屋 | 東海道線 |
|  | 七尾線 | 津幡 | 金沢 | 北陸線 |
|  | 越美北線 | 越前花堂 | 福井 | 北陸線 |
|  | 湖西線 | 近江塩津 | 敦賀 | 北陸線 |
|  | 湖西線 | 山科 | 京都 | 東海道線 |
|  |  | 新大阪 | 大阪 |  |
|  |  | 大阪 | 新大阪 |  |
|  | 福知山線 | 尼崎 | 大阪 | 東海道線 |
|  | 赤穂線 | 東岡山 | 岡山 | 山陽線 |
|  | 伯備線 | 倉敷 | 岡山 | 山陽線 |
|  | 芸備線 | 備中神代 | 新見 | 伯備線 |
|  | 伯備線 | 伯耆大山 | 米子 | 山陰線 |
|  | 瀬戸大橋線 | 宇多津 | 丸亀 | 予讃線 |
|  | 土讃線 | 多度津 | 丸亀 | 予讃線 |
|  | 鳴門線 | 池谷 | 勝瑞 | 高徳線 |
|  | 徳島線 | 佐古 | 徳島 | 高徳線 |
|  | 徳島線 | 佃 | 阿波池田 | 土讃線 |
|  | 予讃線 | 向井原 | 伊予市 | 内子線 |
|  | 予土線 | 北宇和島 | 字和島 | 予讃線 |
|  | 呉線 | ※海田市 | 広島 | 山陽線 |
|  | 可部線 | 横川 | 広島 | 山陽線 |
|  | 山陰線 | 幡生 | 下関 | 山陽線 |
|  | 日豊線 | 西小倉 | 小倉 | 鹿児島線 |
|  | 篠栗線 | 吉塚 | 博多 | 鹿児島線 |
|  | 唐津線 | 久保田 | 佐賀 | 長崎線 |
|  | 日田彦山線 | 城野 | 小倉 | 日豊線 |
|  | 日田彦山線 | 夜明 | 日田 | 久大線 |
|  | 三角線 | 宇土 | 熊本 | 鹿児島線 |
|  | 宮崎空港線 | 田吉 | 南宮崎 | 日南線 |

* 三原以遠 ― 広島間新幹線利用の場合を含む
* 大都市近郊区間は除外(してはならない？？？)

### **データ定義と実装**

t\_lines spe\_route.BSRJCTSP

下図の例において、路線aのdはspe\_route.BSRJCTSP=1とする

この場合、t\_lines.sales\_kmは、c(t.station.rowid)、calc\_kmは、b(t\_line.rowid)が格納されているものとする。また、lflgの分岐駅属性BCJCTFLGもONとなる

**在来線のみ**

d旭川

c新旭川

名寄

上川

滝川

a石北線

b宗谷線

函館線

**新幹線を含む例**

d名古屋

c金山

恵那

三島

米原

a中央西線

b東海道線・新幹線

石北線レコードの旭川はspe\_routeがONである

**実装**

駅選択後、route.add()内で、チェックをおこなう。

<rule1> 路線の着駅がspe\_route のとき

着駅→cに置換

b-dを追加

<rule2> 直前着駅選択時にspe\_route置換(<rule1>適用)があった

if 路線is新幹線 && 新幹線.発駅d-着駅間にcがあるか？

b-dを削除

(最初dだった)cをd(c)に置換

else if 路線==b

b-dを削除（最後のroute\_itemを削除<rule2a>

else

そのまま（指定ルート追加）

※ 最初の路線選択時はなにもスルー

<rule3> 路線-発駅(選択路線の直前着駅)がspe\_route のとき

if 前路線 is 新幹線 && 新幹線-（発駅-駅)間に(c)駅がある

“d(c)”内に置換内部では(c)に

else if 前路線!=(b)

前路線の前に(b－c)を挿入 <rule3a>

else

発駅(直前着駅)を(c)に置換 <rule3b>

route.add()

Rule1適用？

Rule1適用フラグ（Rule2）？

Rule3適用？

この順番とする（たぶん順不同で全チェックで問題なし）。

複合あり得るか？→あり得る

|  |
| --- |
| **a,dから、b, cの取得（t\_linesテーブルより）**  ex.) spe\_route.BSRJCTSP=ONの“石北線-旭川”のレコードの実分岐駅”c新旭川”、通過列車分岐駅の所属路線”bを得る  <入力>  a:分岐路線（石北）  d:乗換駅（旭川）  <出力>  b: sales\_km（本線）(宗谷)→ route.y\_branch\_lineへ保持  c: calc\_km(分岐駅)(新旭川)→ route.y\_branch\_stationへ保持   * spe\_route.BSRJCTSPがOFFのときは、y\_branch\_station、y\_branch\_lineは、0に初期化される(Mustではないはず) |

以下、例に沿って説明する。

#### Ex.1 上川-滝川／恵那-米原

<rule1><rule2>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 上川 | → |  | 上川 |
| 石北a | 旭川d | 石北a | 新旭川c |
| 函館線 | 滝川 | 宗谷b | 旭川d |
|  | | 函館線 | 滝川 |

|  |  |
| --- | --- |
| 旭川を選択 → | 函館線(選択) |
| 富良野線 |
| 宗谷 |

「石北-旭川」を追加時、石北線の旭川がspe\_route.BSRJCTSP=ONなので<rule1>適用として、「石北-旭川」から、y\_branch\_station, y\_branch\_lineを取得し、routeインスタンス変数に保持する。旭川dを新旭川cに置換する。**石北a-旭川dの旭川dの**spe\_route.BSRJCTSP**は、rule1適用フラグをあらわすものとし、RouteItemのflagの**BSRJCTSP bit**もONにする。**

仮に最後に「宗谷b-旭川d」（y\_branch\_line-lineId）を追加する。  
つぎに「函館線‐滝川」を追加時、route\_items[last].flag.BSRJCTSP=1(rule1適用フラグがON)なので<rule2>のチェックとして、函館線(lineId)!=bなので“仮”は仮ではなくそのまま終了。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 恵那 | → |  | 恵那 |  |  | 恵那 |
| 中央西 | 名古屋 | 中央西a | 金山c | 中央西 | 金山c |
| 新幹線 | 米原 | 東海道b | 名古屋d | 東海道 | 米原 |
|  | | 新幹線 | 米原 | (参考:これでもOK) | |

「中央西-名古屋」追加時、<rule1>でd名古屋をc金山に置換、b東海道-d名古屋追加

「新幹線-米原」追加時、<rule2>で新幹線名古屋d～米原間に金山cがないので、そのまま新幹線-米原を追加

#### Ex.2 上川-名寄／恵那‐三島

<rule1><rule2a>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 上川 | → |  | 上川 | → |  | 上川 |
| 石北 | 旭川 | 石北 | 新旭川 | 石北 | 新旭川 |
| 宗谷 | 名寄 | 宗谷 | 旭川 | 宗谷 | 名寄 |
|  | | 宗谷 | 名寄 |  | |

|  |  |
| --- | --- |
| 旭川を選択 → | 函館 |
| 富良野 |
| 宗谷(選択) |

「石北-旭川」を追加時、石北線の旭川がspe\_route.BSRJCTSP=ONなので<rule1>適用として、「石北a-旭川d」から、y\_branch\_station, y\_branch\_lineを取得し、routeインスタンス変数に保持する。旭川d(lineId)を新旭川c(y\_branch\_station)に置換する。石北a-旭川dの旭川dのspe\_route.BSRJCTSPは、rule1適用フラグをあらわすものとし、RouteItem.flag.BSRJCTSP=ONにする。

仮に最後に「宗谷b-旭川d」（y\_branch\_line-lineId）を追加する。  
つぎに「宗谷b‐名寄」を追加時、route\_items[last].flag.BSRJCTSP=1(rule1適用フラグがON)なので<rule2>のチェックとして、宗谷(lineId)=b(y\_branch\_line)なのでroute\_items[last]（仮に追加した「宗谷b-旭川d」(=y\_branch\_line-y\_branch\_station)）を削除してから、lineId, stationId（宗谷b-名寄d）を追加する。

恵那-三島（！新幹線分岐特例）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 恵那 | → |  | 恵那 | → |  | 恵那 |
| 中央西a | 名古屋d | 中央西a | 金山c | 中央西a | 名古屋d(金山c) |
| 新幹線 | 三島 | 東海道b | 名古屋d | 新幹線 | 三島 |
|  |  | 新幹線 | 三島 |  |  |

「中央西a-名古屋d」追加時、<rule1>で名古屋d→金山c変換、東海道b-名古屋d追加

「新幹線-三島」追加時、<rule2>で新幹線なので、新幹線、名古屋d-三島間に金山cがあるので、「新幹線-三島」を追加するまえに、最終アイテムである「東海道b-名古屋d」を削除、金山cを”名古屋d（金山c）”に置換。内部的には、RouteItem.optStationId=名古屋dにする（RouteItem.stationIdは金山cのまま）ことで、表示時、以下のようになる。

RouteItem.optStationId!= 0

分岐駅表示は”stationId.name”

例： 金山

分岐駅表示は  
”optStationId.name(stationId.name)”

例： 名古屋(金山)

通常：

分岐駅特例：

Yes(!=0)

No(=0)

#### Ex.3 滝川‐上川／米原-恵那

<rule3a>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 滝川 | → |  | 滝川 |
| 函館線 | 旭川 | 函館線 | 旭川 |
| 石北 | 上川 | 宗谷 | 新旭川 |
|  | | 石北 | 上川 |

「石北-上川」追加時、石北線乗車駅（t\_lines lineId + route\_items[last].stationId）(=石北-旭川)がspe\_routeの場合(route.add()の引数stationId. BSRJCTFLG=1であるかで決定される)、<rule3>適用となり、「石北-旭川（lineId-route\_items[last].stationId）」から、y\_branch\_station, y\_branch\_lineを取得し、routeインスタンス変数に保持する。前路線（route\_items[last].lineId）函館線!=宗谷(b)(y\_branch\_line)なので、宗谷(y\_branch\_line)-新旭川(y\_branch\_station)(b-c)を追加してから、石北(lineId)-上川(stationId)を追加する。

米原-恵那

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 米原 | → |  | 米原 |  |  | 米原 |
| 新幹線b | 名古屋d | 新幹線 | 名古屋d | 東海道 | 金山 |
| 中央西a | 恵那 | 東海道b | 金山c | 中央西 | 恵那 |
|  | | 中央西 | 恵那 | (参考:これでもOK) | |
|  | |

「中央西-恵那」追加時、<rule3>で、前路線は新幹線で米原-名古屋間に金山がなく、前路線：新幹線!=東海道bで、東海道b-金山cを追加してから「中央西-恵那」を追加<rule3b>

米原-恵那(在来線経由)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 米原 | → |  | 米原 |
| 東海道b | 名古屋d | 東海道b | 金山c |
| 中央西a | 恵那 | 中央西a | 恵那 |

<rule3b>が適用され名古屋を金山に置換

#### Ex.4 名寄-上川／三島-恵那

<rule3b>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 名寄 | → |  | 名寄 |
| 宗谷 | 旭川 | 宗谷 | 新旭川 |
| 石北 | 上川 | 石北 | 上川 |

「石北-上川」追加時、石北線乗車駅（t\_lines lineId + route\_items[last].stationId）(=石北-旭川)がspe\_routeの場合(route.add()の引数stationId. BSRJCTFLG=1であるかで決定される)、<rule3>適用となり、「石北-旭川（lineId-route\_items[last].stationId）」から、y\_branch\_station, y\_branch\_lineを取得し、routeインスタンス変数に保持する。前路線（route\_items[last].lineId）宗谷=宗谷(b)(y\_branch\_line)なので、旭川（route\_items[last].stationId）を新旭川(c)(y\_branch\_station)に置き換え。<rule3b>

三島-恵那（！**新幹線分岐特例）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 三島 | → |  | 三島 | → |  | 三島 |
| 新幹線 | 名古屋d | 新幹線 | 名古屋d | 新幹線 | 名古屋(金山） |
| 中央西a | 恵那 | 東海道b | 金山c | 中央西a | 恵那 |
|  |  | 中央西a | 恵那 |  |  |

※名古屋‐金山間は双方の駅を含め途中下車できません。

「中央西-恵那」追加時、<rule3>で、前路線は新幹線で、三島-名古屋間に金山cがあるので、新幹線降車駅（名古屋d）を金山cにし、optStationIdを名古屋dにする。（表示は「名古屋（金山）」となります）

その後に「中央西-恵那」を追加

最後の条件表示時にも以下の分を追加する。

**「※名古屋‐金山間は双方の駅を含め途中下車できません。」**

#### Ex.5 滝川‐名寄／米原-三島

|  |  |
| --- | --- |
|  | 滝川 |
| 函館線 | 旭川 |
| 宗谷 | 名寄 |

|  |  |
| --- | --- |
|  | 米原 |
| 新幹線 | 名古屋 |
| 東海道 | 三島 |

|  |  |
| --- | --- |
|  | 米原 |
| 東海道 | 名古屋 |
| 新幹線 | 三島 |

|  |  |
| --- | --- |
|  | 米原 |
| 新幹線 | 三島 |

|  |  |
| --- | --- |
|  | 米原 |
| 東海道 | 三島 |

通常ルート：ルール適用無し

#### Ex.6 名寄-滝川／三島-米原

|  |  |
| --- | --- |
|  | 名寄 |
| 宗谷 | 旭川 |
| 函館 | 滝川 |

通常ルート：ルール適用無し

旭川で終わりの場合（分岐選択と着駅選択が事前に明確なため問題ない）

#### Ex.7 上川-旭川／恵那-名古屋

|  |  |
| --- | --- |
|  | 上川 |
| 石北 | 旭川 |

は、<rule1><rule2> で

|  |  |
| --- | --- |
|  | 上川 |
| 石北 | 新旭川 |
| 宗谷 | 旭川 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 恵那 | <rule1>  <rule2>で、→ |  | 恵那 |
| 中央西 | 名古屋 | 中央西 | 金山 |
|  |  | 東海道 | 名古屋 |

#### Ex.8 旭川から上川の例（名古屋から恵那の例）

<rule3a>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 旭川 |  |  | 旭川 |
| 石北 | 上川 | 宗谷 | 新旭川 |
|  | | 石北 | 上川 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 名古屋 |  |  | 名古屋d |
| 中央西 | 恵那 | 東海道b | 金山c |
|  | | 中央西a | 恵那 |

「石北-上川」追加時、石北線乗車駅（t\_lines lineId + route\_items[last].stationId）(=石北-旭川)がspe\_routeの場合(route.add()の引数stationId. BSRJCTFLG=1であるかで決定される)、**<rule3>適用**となり、「石北-旭川（lineId-route\_items[last].stationId）」から、y\_branch\_station, y\_branch\_lineを取得し、routeインスタンス変数に保持する。前路線（route\_items[last].lineId）**NULL**!=宗谷(b)(y\_branch\_line)なので、宗谷(y\_branch\_line)-新旭川(y\_branch\_station)(b-c)を追加してから、石北(lineId)-上川(stationId)を追加する。

#### Ex.9 上川-富良野／恵那-亀山

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 上川 | → |  | 上川 |
| 石北 | 旭川 | 石北 | 新旭川c |
| 富良野線 | 富良野 | 宗谷b | 旭川d |
|  |  | 富良野線 | 富良野 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 恵那 | → |  | 恵那 |
| 中央西 | 名古屋 | 中央西a | 金山c |
| 関西 | 亀山 | 東海道b | 名古屋d |
|  |  | 関西 | 亀山 |

「中央西-名古屋」追加時、<rule1>で名古屋→金山変換、東海道-名古屋追加

「関西-亀山」追加時、<rule2>で亀山!=東海道なのでそのまま関西-亀山を追加

# ■ 新幹線と在来線乗り換え

### **新幹線の分岐駅**

データベース上では、新幹線は並行在来線とは別線として扱う。ただし、並行在来線内の分岐駅もt\_linesに含める。乗換路線リストに表示させないように、sales\_km<0以外をクエリー条件に含めた結果レコードセットのみを表示させる。

|  |
| --- |
| 品川　>  新横浜  　　東神奈川  　　横浜  　　大船  　　茅ヶ崎  　　国府津  小田原 <  熱海  三島 >  　　沼津  　　富士  新富士  静岡 <  京都  　山科+  　草津+  米原 >  岐阜羽島  　大垣  　岐阜  名古屋 <  　金山+  　大府+  三河安城  豊橋  浜松  静岡  > 在来線と別線扱い開始  < 在来線と別線扱い終了  + 路線テーブルに含める  （インデントされた駅） ：新幹線停車駅のない在来線分岐駅 |

### **新幹線と並行在来線の別線扱い区間**

新幹線と在来線が並行している区間では、同じ線として営業キロ等の計算をします。

・東海道・山陽新幹線＝東海道本線・山陽本線・鹿児島本線

・九州新幹線＝鹿児島本線

・東北新幹線＝東北本線

・上越新幹線＝東北本線・高崎線・上越線・信越本線

ただし、次の区間内の各駅（両端の駅を除く）を発駅もしくは着駅または接続駅とする場合は、別の線として営業キロ等の計算をします。

|  |  |
| --- | --- |
| 東海道・山陽新幹線と在来線 | 品川～小田原、三島～静岡、名古屋～米原、新大阪～西明石、福山～三原、三原～広島、広島～徳山 |
| 九州新幹線と在来線 | 博多～久留米、筑後船小屋～熊本 |
| 東北新幹線と在来線 | 福島～仙台、仙台～一ノ関、一ノ関～北上、北上～盛岡 |
| 上越新幹線と在来線 | 熊谷～高崎、高崎～越後湯沢、長岡～新潟 |

### **データ定義と実装**

|  |  |
| --- | --- |
| spe\_route | |
| ビット位置 | 説明 |
| BSRSHINKTRS  BSRSHINKTRSALW | 新幹線駅で平行在来線乗り換え時の侵入許可フラグ  00=通常（三河安城、豊橋、浜松、掛川、熱海、新横浜、東京、新富士など）  01=下り線、平行在来線上り有効（静岡など）  10=上り線、平行在来線下り有効（名古屋、三島など）  11=上下線、平行在来線上下有効（広島など） |

#### Ex.1 下り新幹線（通常駅）→上り在来線

|  |  |  |
| --- | --- | --- |
|  | 東京 |  |
| 東海道新幹線 | 浜松 | 下り |
| 東海道 | 天竜川 | 上り |

新幹線　浜松-掛川は在来線同一視なので後戻りとなり、NG(新幹線下り降車駅浜松のspe\_route. BSRSHINKTRSALW=x0)

#### Ex.2 上り新幹線（通常駅）→上り在来線

|  |  |  |
| --- | --- | --- |
|  | 米原 |  |
| 東海道新幹線 | 浜松 | 上り |
| 東海道 | 天竜川 | 上り |

上りのみなので、OK（後戻りしていない）(新幹線　米原-浜松は上り。東海道　浜松-天竜川も上り)

参考：　新幹線分岐駅：熱海-三島-静岡-掛川-浜松

#### Ex.3 下り新幹線（別線端駅）→上り在来線

|  |  |  |
| --- | --- | --- |
|  | 東京 |  |
| 東海道新幹線 | 静岡 | 下り |
| 東海道 | 草薙 | 上り |

静岡で下り→上り乗車になるが新幹線 静岡のspe\_route. BSRSHINKTRSALW=x1なのでOK

#### Ex.4 上り新幹線（別線端駅）→下り在来線

|  |  |  |
| --- | --- | --- |
|  | 米原 |  |
| 東海道新幹線 | 静岡 | 上り |
| 東海道 | 焼津 | 下り |

静岡で上り→下り乗車になるが新幹線 静岡のspe\_route. BSRSHINKTRSALW=0xなのでNG

#### Ex.5 上り新幹線（別線端駅）→上り在来線

|  |  |  |
| --- | --- | --- |
|  | 名古屋 |  |
| 東海道新幹線 | 静岡 | 上り |
| 東海道 | 草薙 | 上り |

上りのみなので、OK

#### Ex.6 下り在来線→上り新幹線（通常駅）

|  |  |  |
| --- | --- | --- |
|  | 天竜川 |  |
| 東海道 | 浜松 | 下り |
| 東海道新幹線 | 東京 | 上り |

Ex.7の草薙-静岡-東京と同一パターンだが、新幹線 浜松のspe\_route. BSRSHINKTRSALW=x0なのでNG

#### Ex.7 上り在来線→上り新幹線（通常駅）

|  |  |  |
| --- | --- | --- |
|  | 焼津 |  |
| 東海道 | 静岡 | 上り |
| 東海道新幹線 | 東京 | 上り |

上りのみなので、OK（後戻りしていない）(東海道　焼津-静岡。新幹線　静岡-東京は上り。も上り)

#### Ex.8 下り在来線→上り新幹線（別線端駅）

|  |  |  |
| --- | --- | --- |
|  | 草薙 |  |
| 東海道 | 静岡 | 下り |
| 東海道新幹線 | 東京 | 上り |

静岡で下り→上り乗車になるが新幹線 静岡のspe\_route. BSRSHINKTRSALW=x1なのでOK

#### Ex.9 上り在来線→下り新幹線（別線端駅）

|  |  |  |
| --- | --- | --- |
|  | 焼津 |  |
| 東海道 | 静岡 | 上り |
| 東海道新幹線 | 米原 | 下り |

静岡で下り→上り乗車になるが新幹線 静岡のspe\_route. BSRSHINKTRSALW=0xなのでNG

#### Ex.10 上り在来線→上り新幹線（別線端駅）

|  |  |  |
| --- | --- | --- |
|  | 焼津 |  |
| 東海道 | 静岡 | 上り |
| 東海道新幹線 | 東京 | 上り |

上りのみなので、OK

|  |  |  |
| --- | --- | --- |
|  | 神戸 |  |
| 東海道 | 新大阪 | 上り |
| 東海道新幹線 | 東京 | 上り |

|  |  |  |
| --- | --- | --- |
|  | 博多 |  |
| 鹿児島 | 小倉 | 上り |
| 山陽新幹線 | 東京 | 上り |

Route.add(line, station1, station2)

lineは新幹線？

route.add(line, station1, station2)

lineの前路線はline：新幹線の並行在来線？

lineの前路線は新幹線？

Yes

No

end(continue)

下表A

下表B

Yes

No

No

Yes

OK

lineはlineの前路線：新幹線の並行在来線？

No

|  |  |
| --- | --- |
|  |  |
|  | station1 |
| line | station2 |

|  |  |
| --- | --- |
|  |  |
| 新幹線 | station1 |
| line | station2 |

|  |  |
| --- | --- |
|  |  |
|  | station1 |
| line | station2 |

|  |  |
| --- | --- |
|  |  |
|  | station1 |
| line  （新幹線） | station2 |

（表A）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| lineは新幹線で、前路線（routeItem[-1]）は新幹線の並行在来線の場合（Ex.6~Ex.10） | | | | | | | | | |  |
| lineのstation1→station2 | | 前路線（routeItem[-1]）の発駅（routeItem[-2]）→station1 | | | （処理） | | | | |  |
| 下り | | 下り | | | OK | | | | |  |
| 上り | | 上り | | |  |
| 下り | | 上り | | | 前路線着駅（routeItem[-1].station）=station1のspe\_route. BSRSHINKTRSALW=1xの場合？OK ： NG | | | | | a) |
| 上り | | 下り | | | 前路線着駅（routeItem[-1].station）=station1のspe\_route. BSRSHINKTRSALW=x1の場合？OK ： NG | | | | | b) |
| 1. OK | |  |  | | 1. OK Ex.8 |  |  |
|  | | 岐阜 |  | |  | 草薙 |  |
| 東海道 | | 名古屋 | 上り | | 東海道 | 静岡 | 下り |
| 東海道新幹線 | | 米原 | 下り | | 東海道新幹線 | 東京 | 上り |

（表B）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 前路線（routeItem[-1]）は新幹線でlineは前路線の並行在来線の場合（Ex.1~Ex.5） | | | | | | | | | |  |
| 前路線（routeItem[-1]）の発駅（routeItem[-2]）→station1 | | | lineのstation1→station2 | | （処理） | | | | |  |
| 下り | | | 下り | | OK | | | | |  |
| 上り | | | 上り | |  |
| 下り | | | 上り | | 並行在来線の着駅：station2のspe\_route. BSRSHINKTRSALW=1xの場合？OK ： NG | | | | | c) |
| 上り | | | 下り | | 並行在来線の着駅：station2のspe\_route. BSRSHINKTRSALW=x1の場合？OK ： NG | | | | | d) |
| 1. OK |  | |  | | 1. OK Ex.3 |  |  |
|  | 米原 | |  | |  | 東京 |  |
| 東海道新幹線 | 名古屋 | | 上り | | 東海道新幹線 | 静岡 | 下り |
| 東海道 | 岐阜 | | 下り | | 東海道 | 草薙 | 上り |

OK ： 次のチェックへ

NG ： route.add()処理を失敗（-1）で返す。routeItemに追加しない。

#### 新幹線の平行在来線を得る

int getHeikouZairai(line, station1, station2)

line 新幹線

station1 並行在来線駅(新幹線接続駅)

station2 方面

戻り値： =0 並行在来線ではない

≠0 並行在来線line\_id

MSB=1: 上り／0： 下り　←いる？

select\_spe\_route and 0x1ff

from t\_lines

where line\_id = :line -- 新幹線

and station\_id=:station -- の接続駅はspe\_route.b8-0に在来線line\_id

select\_spe\_route　–のbit BSRSHINKTRSALWを取り出す

from t\_lines

where line\_id = :line -- 在来線

and station\_id=:station --新幹線の接続駅

and spe\_route. BSRSHINKTRS=1

# ■ 規則70条　最短経路で計算

下図にしめされる駅を通過する場合、実際の乗車路線に関わらず、最短経路で計算する（重複・復乗禁止）。

発着が近郊区間内駅（新幹線未乗車）相互発着の場合、そちらに準拠する



## 実装概要

大環状線内エリアの進入・脱出路線から、脱出駅、進入駅と、その区間接続最短距離に置き換える。

## データ定義

|  |  |
| --- | --- |
| テーブル　t\_station.city\_flg | |
| ビット | 説明 |
| BCRULE70 | 1ビット: 1で70条適用駅 |

|  |  |  |
| --- | --- | --- |
| テーブル　t\_line. | | |
| フィールド | ビット | 説明 |
| spe\_route | BSR70 | 1ビット: 1で70条定義レコードであることをしめします |
| BSR70MASK | BSR70=1のときのみ有効  14ビット： 進入／脱出路線コード(line\_id) |
| line\_id | テーブル作成時、新規払い出しで、進入路線／脱出路線の2レコード分定義される。経路方向により反転するため、進入路線、脱出路線の区別はデータ上は無い。以下の条件が満たされる  n = (select count(\*) from t\_lines where (spe\_route and (1 << BSR70))!=0  if n/2 = (select count(\*) from t\_lines where (spe\_route and (1 << BSR70)) != 0 group by line\_id | |
| station\_id | 進入／脱出駅 | |
| sales\_km | 1レコード目が0で、2レコード目が進入駅～脱出駅までの営業キロをしめす  (2レコードのいずれか1レコードは0) | |

### **テーブル定義（一部）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| line\_id \*1 | station\_id \*2 | sales\_km | calc\_km | spe\_route \*3 |
| *1001* | 日暮里 | 0 | 0 | 常磐線 |
| *1001* | 錦糸町 | 42 | 0 | 総武線 |
| *1002* | 日暮里 | 0 | 0 | 常磐線 |
| *1002* | 東京 | 58 | 0 | 京葉線 |
| *1003* | 東京 | 0 | 0 | 総武線 |
| *1003* | 東京 | 0 | 0 | 東北新幹線 |
| *1005* | 日暮里 | 0 | 0 | 常磐線 |
| *1005* | 代々木 | 113 | 0 | 中央東線 |
| : | : | : | : | : |

\*1 仮の数値（1000からインクリメント生成）

\*2 実際はid値

\*3 ビットBSR70=1

11路線（N×（N-1）とおり）の計110レコード（55グループ=line\_id）の定義（以降のマトリクス参照）

* 但し、新幹線→在来線のように戻るために重複ルートになりあり得ないものもあるのは定義しないので110レコードは無い

上記テーブル定義でline\_id=3のグループはstation\_idが同一となり、DBユニーク制約に違反するため、station\_idのMSB(bit31)をONにする

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *1003* | 東京 | 0 | 0 | 総武線 |
| *1003* | 東京 | 0 | 0 | 東北新幹線 |

* line\_idは、t\_lineテーブルの外部参照だが架空のIDを挿入することがSQLiteでは可能なのでとくにt\_lineテーブル側にダミー路線名レコードを挿入しない方式とする（NGなDBシステムの場合、!ではじまる路線名定義や、#ではじまる路線名定義でダミー路線レコードを挿入することで対応する。

### **クエリー**

select t1.line\_id,

t1.station\_id,

t2.station\_id,

t1.spe\_route and NOT(1 << BSR70),

t2.spe\_route and NOT(1 << BSR70),

from t\_lines t1 join t\_lines t2

on t1.line\_id=t2.line\_id and

(t1.spe\_route and 0x3fff)<>(t2.spe\_route and 0x3fff)

where (t1.spe\_route and 0x3fff)='中央東線' and

(t2.spe\_route and 0x3fff)='常磐線';

* SQLはシンボルが利用できないので、BSR70を置き換える必要あり

上記クエリー例：

侵入路線：中央東線→脱出路線：常磐線の場合、t1.station\_idに、新宿、t2.station\_idに日暮里が返される。t1.line\_idは、以降の例で、l\_rule70に置き換える

中央東線、常磐線でクエリーを実行すると、

「line\_id, 新宿、日暮里, 中央東線, 常磐線」

が返される

常磐線、京葉線でクエリーを実行すると、

「line\_id,日暮里,東京,常磐線,京葉線」

が返される

### **路線間接続駅の定義**

2つの接続駅の最短営業キロを定義する。

接続駅が1つの場合、入口、出口とも同一駅で、営業キロ=0kmとして定義・計算する

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 常磐線 | 総武線 | 京葉線 | 東北線 | 東北新幹線 | 中央東線 | 上越新幹線 | 長野北陸新幹線 | 東海道線 | 東海道新幹線 |
| 常磐線 |  |  |  |  |  |  |  |  |  |  |
| 総武線 | 日暮里  錦糸町 |  |  |  |  |  |  |  |  |  |
| 京葉線 | 日暮里  東京\*1 | 東京  東京 |  |  |  |  |  |  |  |  |
| 東北線 | 日暮里 | 錦糸町  秋葉原 | 東京\*1 |  |  |  |  |  |  |  |
| 東北線（埼京線） | 日暮里  赤羽 | 錦糸町  赤羽 | 東京  赤羽 |  |  | 新宿  赤羽 |  |  | 東京  赤羽 | 東京  赤羽 |
| 東北新幹線 | 日暮里 | 錦糸町  秋葉原 | 東京\*1 |  |  |  |  |  |  |  |
| 中央東線 | 日暮里  新宿 | 錦糸町  御茶ノ水 | 東京\*2 | 赤羽  新宿 | 赤羽  新宿 |  |  |  |  |  |
| 上越新幹線 | 日暮里 | 錦糸町  秋葉原 | 東京\*1 |  |  | 新宿  赤羽 |  |  |  |  |
| 長野北陸新幹線 | 日暮里 | 錦糸町  秋葉原 | 東京\*1 |  |  | 新宿  赤羽 |  |  |  |  |
| 東海道線 | 日暮里  東京 | 錦糸町  東京 | 東京\*1 | 東京 | 東京 | 代々木  品川 | 東京 | 東京 |  |  |
| 東海道新幹線 | 日暮里  東京 | 錦糸町  東京 | 東京\*1 | 東京 | 東京 | 代々木  品川 | 東京 | 東京 |  |  |

\*1 ・・・ 京葉線・蘇我以遠の場合、外房線・千葉〜総武線・錦糸町、秋葉原、日暮里経由

\*2 ・・・ 京葉線・蘇我以遠の場合、外房線・千葉〜総武線・錦糸町、お茶の水経由

※ 総武線⇔京葉線についえは、蘇我以遠はあり得ない。あるとすると両線の発着駅は、蘇我以近に限られるため、規則69条の適用もあり得ない。

東海道新幹線⇔東海道線、東北新幹線⇔上越新幹線などが定義されていないのは、あり得ないため（あり得たらバグ）

## 実装

運賃計算時に判定する。

* 本70条適用後、69条の適用後、さらに本70条適用（常磐‐京葉など）

|  |  |  |
| --- | --- | --- |
|  | 適用パターン | |
| A |  | 対象外（86、87条適用） |
| B |  | 対象外 |
| **C** |  | **適用** |
| D |  | 対象外（86、87条適用） |
| E |  | 対象外（86、87条適用） |
| F |  | 対象外（○内最短経路） |
| G |  | **対象外(86、87条も非適用)** |
| H |  | **対象外(86、87条も非適用)** |

#### 営業規則70条のチェック

int checkOfRule70(Route& route)；

Route配列内の70条チェックをおこなう。

戻り値は、算出した最短経路の営業キロの総和（都区内なので計算キロは無し）

a70\_sales\_km = 0

f70pass = f70 = false

le = lb = sb = se = 0

70適用駅

f70 = false

f70 = true

f70 = true

路線→lb

駅→sb

70「非」適用駅

f70 = false

路線→le

駅→se

a70\_sales\_km +=

(sb~se間営業キロ算出)

Route配列（発駅～着駅）

f70pass = true

f70pass = true

70非適用駅

G, H適用時(非採用)

a70\_sales\_km = 0

f70pass = f70 = false

le = lb = sb = se = 0

70適用駅

f70 = false

f70 = true

f70 = true

路線→lb

駅→sb

70「非」適用駅

f70 = false

路線→le

駅→se

a70\_sales\_km +=

(sb~se間営業キロ算出)

Route配列（発駅～着駅）

f70pass = true

f70pass = true

70非適用駅

f70pass = false

発駅=70条適用駅(E、G)

#### 東海道-中央

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 計算経路 | |  | データ構造 | |
|  | 川崎 |  |  | 川崎 |  |  | 川崎 |
| **東海道** | **東京** |  | 東海道 | 品川 |  | 東海道 | **品川** |
| **中央東** | **八王子** |  | 山手 | 代々木 |  | **l\_rule70** | **代々木** |
|  |  |  | 中央 | 八王子 |  | 中央 | 八王子 |

sb: 東京／lb: 東海道／le:中央東／se:八王子

qry： 品川、代々木

l\_rule70には、クエリー結果のt\_lines.line\_idがはいる。

＜処理フロー＞

1. t\_linesテーブルより、spe\_route.BSR70 ON and (「東海道」 or 「中央東」)でクエリ－を実行し、「品川」、「代々木」とline\_idを得る。
2. Route配列のsbのインデックスの駅を「品川」に、seのインデックスに配列を挿入し、その駅を「代々木」とする。路線はline\_idにする。
3. Route配列のsbのインデックス+1～seのインデックス-2の配列は削除する（あれば）。
4. 重複駅の削除
5. 以上で終了

以下例をしめす

凡例

最左段太字： sb,lb / se/ le

最右段：b: Index of lb,sb

最右段：e: Index of le, se

qry： lb, leでのクエリー-結果

l\_rule70: line\_idが入る

(delete) : 削除レコード

~~取り消し線~~ ： 削除レコード

#### 中央-東海道

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 計算経路 | |  | 70条適用後データ構造 | |  |
|  | 八王子 |  |  | 八王子 |  |  | 八王子 |  |
| **中央東** | **東京** |  | 中央東 | 代々木 | b | 中央東 | 代々木 |  |
| **東海道** | **川崎** |  | 山手 | 品川 |  | l\_rule70 | 品川 |
|  |  |  | 東海道 | 川崎 | e | 東海道 | 川崎 |  |

qry: 代々木、品川

#### 東北-京葉（蘇我以遠）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 70条→69条→157条適用後経路 | |  | 70条適用後データ構造 | |  |
|  | 宇都宮 |  |  | 宇都宮 |  |  | 宇都宮 |  |
| **東北** | **田端** |  | 東北 | 秋葉原 | b | 東北 | **東京** |  |
| 山手 | 代々木 |  | 総武線(お茶の水-錦糸町) | 錦糸町 |  | **~~l\_rule70~~** | **~~東京~~** |  |
| 中央東 | 東京 |  | 総武線 | 千葉 | e | 京葉 | 蘇我 |  |
| **京葉** | **蘇我** |  | 外房線 | 蘇我 |  |  |  |  |

qry: 東京、東京

* このあとの69条の置換で上記中央段の経路に置き換わる（東京経由の総武線）
* さらに再度、本70条の適用後、157条2項適用で上記中央段の経路となる

#### 常磐-京葉（蘇我以前）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 計算経路 | |  | 70条適用後データ構造 | |  |
|  | いわき |  |  | いわき |  |  | いわき |  |
| **常磐** | **日暮里** |  | 常磐 | 日暮里 | b | 常磐 | **日暮里** |  |
| 東北 | 東京 |  | 東北 | 東京 |  | **l\_rule70** | **東京** |
| **京葉** | **葛西** |  | 京葉 | 葛西 | e | 京葉 | 葛西 |  |

qry: 日暮里、東京

#### 常磐-京葉（蘇我以遠）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 70条→69条適用後経路 | |  | 70条適用後データ構造 | |  |
|  | いわき |  |  | いわき |  |  | いわき |  |
| **常磐** | **日暮里** |  | 常磐 | 日暮里 | b | 常磐 | **日暮里** |  |
| 東北 | 東京 |  | 東北 | 秋葉原 |  | **l\_rule70** | **東京** |
| **京葉** | **蘇我** |  | 総武線(お茶の水-錦糸町) | 錦糸町 | e | 京葉 | 蘇我 |  |
|  |  |  | 総武線 | 千葉 |  |  |  |  |
|  |  |  | 外房線 | 蘇我 |  |  |  |  |

qry: 日暮里、東京

※ このあとの69条の置換で上記中央段の経路に置き換わる（東京経由の総武線）

* さらに再度、本70条の適用後、157条2項適用で上記中央段の経路となる

#### 常磐-上越新幹線1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 計算経路 | |  | 70条適用後データ構造 | |  |
|  | いわき |  |  | いわき |  |  | いわき |  |
| **常磐** | **日暮里** |  | 常磐 | 日暮里 | b | 常磐 | **日暮里** |  |
| 東北 | 東京 |  | 上越新幹線 | 高崎 |  | **~~l\_rule70~~** | **~~日暮里~~** |  |
| **上越新幹線** | **高崎** |  |  |  | e | 上越新幹線 | 高崎 |  |

qry: 日暮里、日暮里

* 上野、東京は戻る（乗車可）
* 日暮里-東京-上越新幹線は重複経路で指定不可

#### 常磐-上越新幹線2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 計算経路 | |  | 70条適用後データ構造 | |  |
|  | いわき |  |  | いわき |  |  | いわき |  |
| **常磐** | **日暮里** |  | 常磐 | 日暮里 | l | 常磐 | **日暮里** |  |
| 東北 | 品川 |  | 上越新幹線\* | 高崎 |  | **~~l\_rule70~~** | **~~日暮里~~** |  |
| 山手 | 池袋 |  |  |  | e | 上越新幹線 | 高崎 |  |
| 赤羽 | 赤羽 |  |  |  |  |  |  |  |
| 埼京 | 大宮 |  |  |  |  |  |  |  |
| **上越新幹線** | **高崎** |  |  |  |  |  |  |  |

qry: 日暮里、日暮里

#### 京葉（蘇我以近）-中央

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 計算経路 | |  | 70条適用後データ構造 | |  |
|  | 葛西 |  |  | 葛西 |  |  | 葛西 |  |
| **京葉** | **東京** |  | 京葉 | 東京 | s | 京葉 | **東京** |  |
| **中央東** | **八王子** |  | 中央東 | 八王子 |  | **~~l\_rule70~~** | **~~東京~~** |  |
|  |  |  |  |  | e | 中央東 | 八王子 |  |

qry: 東京、東京

#### 京葉（蘇我以遠）-中央

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 70条→69条→157条適用後経路 | |  | 70条適用後データ構造 | |  |
|  | 蘇我 |  |  | 蘇我 |  |  | 蘇我 |  |
| **京葉** | **東京** |  | 外房 | 千葉 | b | 京葉 | **東京** |  |
| **中央東** | **八王子** |  | 総武 | 錦糸町 |  | **~~l\_rule70~~** | **~~東京~~** |  |
|  |  |  | 総武(錦糸町-御茶ノ水) | 御茶ノ水 | e | 中央東 | 八王子 |  |
|  |  |  | 中央東 | 八王子 |  |  |  |  |

qry: 東京、東京

※ このあとの69条の置換で上記中央段の経路に置き換わる（東京経由の総武線）

* さらに再度、本70条の適用後、157条2項適用で上記中央段の経路となる

#### 総武（蘇我以遠）－京葉

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 計算経路 | |  | 70条適用後データ構造 | |  |
|  | 大網 |  |  | 大網 |  |  | 大網 |  |
| 外房 | 千葉 |  | 外房 | 千葉 |  | 外房 | 千葉 |  |
| **総武** | **錦糸町** |  | 総武 | 東京 | b | 総武 | **東京** |  |
| 総武(錦糸町-御茶ノ水) | 秋葉原 |  | 京葉 | 蘇我 |  | (delete) | |  |
| 東北 | 東京 |  |  |  |  | **~~l\_rule70~~** | **~~東京~~** |  |
| **京葉** | **蘇我** |  |  |  | e | 京葉 | 蘇我 |  |

qry: 東京、東京

* このあと、規則69条が適用されない。
* 規則69条を適用させ、往復とも同じ経路になり、分割購入（大網-東京、東京-蘇我）の方が\40安い

#### 常磐-東北（埼京線）1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 計算経路 | |  | 70条適用後データ構造 | |  |
|  | いわき |  |  | いわき |  |  | いわき |  |
| **常磐** | **日暮里** |  | 常磐 | 日暮里 | b | 常磐 | **日暮里** |  |
| 東北 | 品川 |  | 東北 | 赤羽 |  | (delete) | |
| 山手 | 池袋 |  | 埼京 | 武蔵浦和 |  | (delete) | |
| 赤羽 | 赤羽 |  | 武蔵野 | 西国分寺 |  | **l\_rule70** | **赤羽** |
| **埼京** | **武蔵浦和** |  |  |  | e | 埼京 | 武蔵浦和 |  |
| 武蔵野 | 西国分寺 |  |  |  |  | 武蔵野 | 西国分寺 |  |

qry: 日暮里、赤羽

#### 常磐-東北（埼京線）2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 乗車経路 | |  | 70条→69条→157条適用後経路 | |  | 70条適用後データ構造 | |  |
|  | いわき |  |  | いわき |  |  | いわき |  |
| **常磐** | **日暮里** |  | 常磐 | 日暮里 | b | 常磐 | **日暮里** |  |
| 東北 | 品川 |  | 東北 | 中浦和 |  | (delete) | |
| 山手 | 池袋 |  | 武蔵野 | 新松戸 |  | (delete) | |
| 赤羽 | 赤羽 |  |  |  |  | **l\_rule70** | **赤羽** |
| **埼京** | **武蔵浦和** |  |  |  | e | 埼京 | 武蔵浦和 |  |
| 武蔵野 | 新松戸 |  |  |  |  | 武蔵野 | 新松戸 |  |

qry: 日暮里、赤羽

# ■ 規則69条最短経路で計算

## 規則69条

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **乗車経路** | | | **計算経路** | | |
| **発** | **路線** | **至** | **計算路線** | **発** | **至** |
| 1 | 大沼 | 函館線(東森経由) | 森 | 函館線 | 大沼 | 森 |
| 2 | 日暮里 | 東北線(尾久経由) | 赤羽 | 東北線 | 日暮里 | 赤羽 |
| 3 | 赤羽 | 東北線(埼京線) | 大宮 | 東北線 | 赤羽 | 大宮 |
| 4 | 品川 | 東海道線(西大井経由) | 鶴見 | 東海道線 | 品川 | 鶴見 |
| 5 | 東京 | 京葉線 | 蘇我 | 総武線 | 東京 | 千葉 |
| 外房線 | 千葉 | 蘇我 |
| 6 | 山科 | 東海道線 | 米原 | 湖西線 | 山科 | 近江塩津 |
| 米原 | 北陸線 | 近江塩津 |
| \* | 大阪 | 大阪環状線  （天満経由） | 天王寺 | 大阪環状線  （福島経由） | 大阪 | 天王寺 |
| 7 | 三原 | 呉線 | 海田市 | 山陽線 | 三原 | 海田市 |
| 8 | 岩国 | 山陽線 | 櫛ヶ浜 | 岩徳線 | 岩国 | 櫛ケ浜 |
| 9 | 秋葉原 | 東北線 | 東京 | 総武線(錦糸町-御茶ノ水) | 秋葉原 | 錦糸町 |
| 東京 | 総武線 | 錦糸町 |
| 10 | 御茶ノ水 | 中央東線 | 神田 | 総武線(錦糸町-御茶ノ水) | 御茶ノ水 | 錦糸町 |
| 神田 | 東北線 | 秋葉原 |
| 10 | 御茶ノ水 | 中央東線 | 東京 | 総武線(錦糸町-御茶ノ水) | 御茶ノ水 | 錦糸町 |
| 東京 | 総武線 | 錦糸町 |
|  |  |  |  |  |  |  |

\* 大阪環状線は大阪-天王寺は計算時必然的に最短で計算されるので特別処理は不要

※ 「御茶ノ水」（中央東線）「東京」（東北線）「秋葉原」は、東京-秋葉原間に神田が存在するため指定できない

#### 1. 大沼-森

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 函館 |  |  | 函館 |
| 函館線 | **大沼** |  | ↓ | ↓ |
| **函館線（東森経由）** | **森** |  | ↓ | ↓ |
| 函館線 | 長万部 |  | 函館線 | 長万部 |

大沼

森

東森

大沼公園

函館

長万部

函館線

函館線

検索： 函館線（東森経由）、大沼、森

|  |  |
| --- | --- |
|  | 函館 |
| 函館線 | 長万部 |

そのまま

#### 2. 日暮里-赤羽

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 盛岡 |  |  | 盛岡 |
| **東北線（日暮里経由）** | **赤羽** |  | ↓ | ↓ |
| 東北線 | **日暮里** |  | ↓ | ↓ |
| 東北線 | 東京 |  | 東北線 | 東京 |

検索： 東北線（日暮里経由）、赤羽、日暮里

#### 3. 赤羽-大宮

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 東京 |  |  | 東京 |
| 東北線 | **赤羽** |  | ↓ | ↓ |
| **東北線（埼京線）** | **大宮** |  | ↓ | ↓ |
| 東北線 | 盛岡 |  | 東北線 | 盛岡 |

検索： 東北線（埼京線）、赤羽、武蔵浦和、大宮

#### 4. 品川-鶴見

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 東京 |  |  | 東京 |
| 東海道線 | **品川** |  | ↓ | ↓ |
| **東海道線（西大井経由）** | **鶴見** |  | ↓ | ↓ |
| 東海道線 | 横浜 |  | 東海道線 | 横浜 |

検索： 東海道線（西大井経由）、品川、武蔵小杉、鶴見

#### 5. 東京-蘇我

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **東京** |  |  | 東京 |  |
| **京葉線** | **蘇我** |  | 総武線 | 千葉 |  |
| 外房線 | 勝浦 |  | 外房線 | 勝浦 | 蘇我 |

錦糸町

蘇我

総武線

東京

京葉線

外房線  
（千葉-安房鴨川）

千葉

武蔵野線

内房線

検索： 京葉線、東京、市川塩浜、南船橋、蘇我

#### 6. 山科-近江塩津

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 大阪 | 山科 |  | 大阪 |
| 東海道線 | 米原 |  | 東海道線 | 山科 |
| 北陸線 | 福井 | 近江塩津 | 湖西線 | 近江塩津 |
|  |  |  | 北陸線 | 福井 |

山科

敦賀

米原

近江塩津

大阪・京都

名古屋・東京

湖西線

東海道線

草津

北陸線

検索： 東海道線、山科、草津、米原

北陸線、米原、近江塩津

#### 7. 三原-海田市

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 尾道 |  |  | 尾道 |
| 山陽線 | **三原** |  | ↓ | ↓ |
| 呉線 | **海田市** |  | ↓ | ↓ |
| 山陽線 | 広島 |  | 山陽線 | 広島 |

海田市

三原

呉

西条

広島

尾道

山陽線

呉線

山陽線

山陽線

検索： 呉線、海田市、三原

#### 8. 岩国-櫛ヶ浜

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 広島 |  |  | 広島 |
| ↓ | ↓ | 岩国 | 山陽線 | 岩国 |
| ↓ | ↓ | 櫛ヶ浜 | 呉線 | 櫛ヶ浜 |
| 山陽線 | 新山口 |  | 山陽線 | 新山口 |

櫛ヶ浜

岩国

柳井

徳山

広島

岩徳線

山陽線

山陽線

山陽線

検索： 山陽線、櫛ヶ浜、岩国

## 実装概要

実装法；

規則70条、86、87条のあとに行う

運賃計算時、経路検索して、置き換える。

このとき、運賃計算は毎回行っているので、MAX\_A69ROUTELINES<経路配列数のときは、経路配列数- MAX\_A69ROUTELINESから検査をする（前半のルートは置換済みであるので検索から除外）

MAX\_A69ROUTELINES=3

以下の注釈を「注意事項に含める」

「岩国-櫛ヶ浜間は、岩徳線経由でも山陽線（新幹線含む）経由で乗車可能」

## データ定義

### **データ定義条件**

* spe\_route. BSR69TERM =1は、営業規則69条の端駅に定義（上記表の「発」、「至」）
* 1路線にspe\_route.BSR69TERM=1かつ同一のspe\_route.BSR69NOMASKが2つあると置換する
* 1路線にspe\_route.BSR69TERM=1が1つもない場合は、営業規則69条に該当しないのでなにもしない。
* 1路線にspe\_route.BSR69TERM=1が1つしかない場合は、「発」～「至」駅の途中駅から経路を外れているので営業規則69条には該当せずなにもしない。
* 1路線にspe\_route.BSR69TERM=1が3つ以上存在することはない（現状はないが将来あり得ないこともないが、spe\_route.BSR69NOMASKまでは同一ではない）
* 1路線にspe\_route.BSR69TERM=1で異なるspe\_route. BSR69NOMASKを挟むことはない（必ずspe\_route.BSR69TERM=1+spe\_route. BSR69NOMASKは連続する）。
* spe\_route.BSR69TERM=1は複数路線をまたぐ場合があり得る。この場合、乗り換えて継続する駅をspe\_route.BSR69CONT=1でしめす（上表では「米原」にのみ該当）。
* 一周ルートは適用させないように（総武-京葉ルート大網-蘇我-東京-蘇我や、尾道～三原～呉線～海田市～山陽線～西条への切符）  
  さらに総武線のときは、秋葉原経由、東京経由双方とも考慮されている必要がある

### **テーブル定義： t\_rule69**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | station\_id1 | station\_id2 | line\_id | ord |
| 1 | 大沼 | 森 | 函館線 | 0 |
| 2 | 日暮里 | 赤羽 | 東北線 | 0 |
| 3 | 赤羽 | 大宮 | 東北線 | 0 |
| 4 | 品川 | 鶴見 | 東海道線 | 0 |
| 5 | 東京 | 千葉 | 総武線 | 0 |
| 5 | 千葉 | 蘇我 | 外房線 | 1 |
| 6 | 山科 | 近江塩津 | 湖西線 | 0 |
| 7 | 三原 | 海田市 | 山陽線 | 0 |
| 8 | 岩国 | 櫛ヶ浜 | 岩徳線 | 0 |

## 実装詳細

### **検索処理**

spe\_route.BSR69NOMASK

=a69id

a69id

=0

≠0

路線、発駅～着駅でspe\_route.BSR69TERM=1 order by sales\_km

（路線、発駅、着駅）（発駅は配列のインデックス-1）

a69id ← spe\_route.BSR69NOMASK

駅保持→a69list.station1

≠a69id

a69id ←0

spe\_route.BSR69TERM=1が2つ検出(3つ以上はあり得ない)

Route配列の次の路線の発駅～着駅で以下の処理

spe\_route.BSR69CONT=1

**置換処理**

N

Y

spe\_route.BSR69TERM=1の1つめ

a69id ←0

路線→a69list.line、（路線の着駅の）route配列インデックス保持

路線、発駅～着駅でspe\_route.BSR69TERM=1 order by sales\_km

spe\_route.BSR69NOMASK

=a69id

≠a69id

=0

>0

cnt ← 0

a69list index++

路線→a69list.line

cnt

発駅？

cnt++

a69list.station1

spe\_route.BSR69CONT=1

N

Y

**置換処理**

着駅と一致？

Route配列の次有

Route配列の次の路線の発駅～着駅で再帰処理

着駅と一致？

Route配列の次有

a69list ：line, station1, station2の3つのint16の配列でMAX\_A69ROUTELINESを静的確保

駅保持：→a69list.station2

駅保持：→a69list.station2

* a69list配列は、上表の「乗車経路」の「路線」「発」「至」を保持されることになります。  
  a69list配列は、3つのメンバすべて≠0で有効であり、1つでも=0があるレコードは無効で終端を表します

**上記フローを一部クエリー化**

**HCP前段をクエリー化＞引数：line\_id, station\_id1, station\_id2（路線の駅１〜駅2間にBSR69TERM!=0の駅を抽出）＞出力：a69list.station, a69list.lflg(該当駅とその属性を返す)（((lflg>>17)&0x0f)レコードが連続して2行返されること）**

**連続して2行に満たない場合、途中下車、乗り換えが考えられるのでチェック対象外**

**連続して2行存在していた場合、置換処理へ、**

**さらにそのまえにa69list.stationの2レコード目の((lflg&(1<<BSR69TERM)!=0の場合（米原駅を指す）、station\_id2が一致しているか判定。不一致の場合、上りの場合、東海道線を名古屋方面へ行ってしまっていている（下りは湖西線の終点が米原であるためありえない）ので、置換適用外。一致していた場合、RouteListの次のレコードのlineIdのstationIdからつぎのstationIdまでの間にBSR69TERM!=0かつ((lflg>>17)&15)!=0の駅がある場合、**

**(→：クエリー結果a69list)**

呉線型

[呉線]、三原―海田市

→呉線、三原、海田市

岩徳線型

[山陽線]、新山口（－櫛ヶ浜―岩国）―広島

→山陽線、櫛ヶ浜、岩国

米原型

[東海道線]、京都（－山科―）米原－[北陸線]―（近江塩津）―福井

→東海道線、山科、米原

北陸線、米原、近江塩津

京葉線型

[京葉線]、東京―蘇我ー[外房線]－勝浦

→京葉線、東京、蘇我

### **置換処理**

置換処理は3パターン存在する。以下にパターンについてしめす。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 乗車経路 | 計算経路 | 例 |
| 置換処理1 | 1 | 1 | 岩国－櫛ヶ浜（山陽線、岩徳線）、三原-海田市（呉線、山陽線）など多数 |
| 置換処理2 | 1 | N | 東京‐蘇我（京葉線、総武線・外房線） |
| 置換処理3 | N | 1 | 山科‐近江塩津（東海道線・北陸線、湖西線） |
| － | N | N | 例無し（実装しない）※ |
| 置換処理未実施 | ― | ― | 計算経路（または乗車経路）を経て乗車経路（または計算経路）に乗車する例では置換はおこなわない。  例： 外房線・大網から千葉、総武線を経て東京から京葉線で蘇我へ至るルートの場合、京葉線・東京－蘇我間の置換はおこなわない  （但し、上記例の片道乗車券より大網－東京、東京－蘇我という乗車券を分割購入した方が\40‐安い） |

* 現在Nは2までしかないので実装は2までの対応とする。また、国鉄・JRグループの過去50年を振り返っても今後も縮小傾向にあって将来増えることは考えにくい。

下図は、上記3パターンの判定処理。パターン毎の処理詳細については以降にしめします。

t\_rule69をid列で抽出（SQL）

置換処理（路線、駅1、駅2、Route配列インデックス）：Idx

配列[MAX\_A69ROUTELINES]

t\_rule69行数==1

「置換処理(1/3)」

「置換処理(3/3)」

Y

Y

N

N

a69list  
行数==1

a69list  
行数==1

NotImplement（Failure）

「置換処理(2/3)」

N

Y

lineDupCheck2(t\_rule69[], Route)

a69list： 引数の路線、駅1、駅2

OK

NG

return

#### lineDupCheck2() – 69条置換判定

（総武-京葉ルート大網-蘇我-東京-蘇我や、尾道～三原～呉線～海田市～山陽線～西条への切符）

lineDupCheck2(t\_rule69 [] ritem, Route route)

route[N] (N=1~route.Length-1)

→ route[N].line:line, route[N-1].station:station1, route[N].station:station2

1 < route.length

ritem[N] (N=0~ritem.Length-1)→

ritem[N].line:line69, ritem[N].station1:station691, ritem[N].station2:station692

line == line69

t\_linesクエリー- line and station1 or station2 or station691 or station692 order by sales\_km

結果配列数3未満 or 5以上(あり得ない)

return NG:置換しない

結果配列数3 or 4

結果配列先頭よりstation691 or station692を探す→ idx69

if ( ! (結果配列[idx69]==station691 &&   
結果配列[idx69+1]==station692) or   
(結果配列[idx69]==station692 &&  
結果配列[idx69+1]==station691))

dupflg = false(OK:置換する)

return OK:置換する

置換経路

他路線

置換しない

置換する

置換する

置換しない

乗車（経路）路線（Route配列内）

t\_rule69駅1、駅2(station691, station692)

rule69駅1、駅2の間に乗車駅があれば置換しない

**（上記フローは破棄）**

**全Routeで、route[N].line == t\_rule69.lineがあった場合t\_rule69.staion1～t\_rule69.station2内に、route[N-1].stationと、route[N].stationのいずれかがあった場合（inStation()）、（但しいずれか一方の場合、その一方がt\_rule69の端駅と一致の場合置換する）置換適用外とする。**

**↓↓↓↓**

**全Routeで、route[N].line == t\_rule69.lineがあった場合t\_rule69.staion1～t\_rule69.station2内に、route[N-1].stationと、route[N].stationの2駅ともあった場合（inStation()）、置換適用外とする（置換しない）。**

**→ 錦糸町-御茶ノ水-東京-・・・の例があるので適用していた。**

#### 置換例1 三原→海田市（下り）

a69list

|  |  |  |
| --- | --- | --- |
| 呉線 | 三原 | 海田市 |

t\_rule69 recordset SQL

|  |  |  |
| --- | --- | --- |
| 山陽線 | 三原 | 海田市 |

a) 外→外

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *神戸* | ⇒ |  |  | *神戸* | ⇒ |  | *神戸* |
|  | *山陽線* | 三原 |  | *山陽線* | 三原 | *山陽線* | *下関* |
| → | 呉線 | 海田市 | → | **山陽線** | 海田市 |  |  |
|  | *山陽線* | *下関* |  | *山陽線* | *下関* |  |  |

b) 内→外

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 三原 | ⇒ |  |  | 三原 | ⇒ |  | 三原 |
| → | 呉線 | 海田市 | → | **山陽線** | 海田市 | 山陽線 | *下関* |
|  | *山陽線* | *下関* |  | *山陽線* | *下関* |  |  |

c) 内→内

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 三原 | ⇒ |  |  | 三原 |  |  |  |
| → | 呉線 | 海田市 | → | **山陽線** | 海田市 |  |  |

d) 外→内

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *神戸* | ⇒ |  |  | *神戸* | ⇒ |  | *神戸* |
|  | *山陽線* | 三原 |  | *山陽線* | 三原 | 山陽線 | 海田市 |
| → | 呉線 | 海田市 | → | **山陽線** | 海田市 |  |  |

→ インデックスの指す路線のみを置きかえ

#### 置換例2 下関→神戸（櫛ヶ浜、岩国）(上り)

a69list

|  |  |  |
| --- | --- | --- |
| 山陽線 | 櫛ヶ浜 | 岩国 |

t\_rule69 recordset SQL

|  |  |  |
| --- | --- | --- |
| 岩徳線 | 岩国 | 櫛ヶ浜 |

a) 外→外上り

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 下関 |  |  | 下関 |  |
| → | 山陽線 | 神戸 |  | **山陽線** | **櫛ヶ浜** |  |
|  |  |  |  | **岩徳線** | **岩国** |  |
|  |  |  | → | 山陽線 | 神戸 |  |

b) 内→外

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 櫛ヶ浜 |  |  | 櫛ヶ浜 |  |
| → | 山陽線 | 神戸 |  | **岩徳線** | **岩国** |  |
|  |  |  | → | 山陽線 | 神戸 |  |

（適用外）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 下関 |  |  | 下関 |  |
|  | 山陽線 | 櫛ヶ浜 |  | 山陽線 | 櫛ヶ浜 | 適用外 |

c) 内→内

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 櫛ヶ浜 |  |  | 櫛ヶ浜 |  |
| → | 山陽線 | 岩国 | → | **岩徳線** | 岩国 |  |

d) 外→内

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 下関 |  |  | 下関 |  |
| → | 山陽線 | 岩国 |  | **山陽線** | **櫛ヶ浜** |  |
|  |  |  | → | **岩徳線** | 岩国 |  |

a) 外→外下り

a69list

|  |  |  |
| --- | --- | --- |
| 山陽線 | 岩国 | 櫛ヶ浜 |

t\_rule69 recordset SQL

|  |  |  |
| --- | --- | --- |
| 岩徳線 | 岩国 | 櫛ヶ浜 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 神戸 |  |  | 神戸 |  |
| → | 山陽線 | 下関 |  | **山陽線** | **岩国** |  |
|  |  |  |  | **岩徳線** | **櫛ヶ浜** |  |
|  |  |  | → | 山陽線 | 下関 |  |

#### 置換処理(1/3) ：乗車路線(1線)計算路線(1線)に限る実装(a69list is 1 / t\_rule69 is 1)

乗車駅（経路分岐駅かその外か）、路線（乗車 or 下車駅路線=乗車経路路線（置換例2）／乗車 or下車路線≠乗車経路路線（置換例1））によって処理が異なる（上り下りとも同一処理で可）。

置換例1

海田市

三原

呉

西条

広島

尾道

山陽線

呉線

山陽線

山陽線

乗車路線

計算路線

置換例2

櫛ヶ浜

岩国

柳井

徳山

広島

岩徳線

山陽線

山陽線

山陽線

計算経路

Route[Idx].駅==a69list.駅2

yes : 置換例1 or c) or d)

* 配列インデックスの’-1’はラストインデックスをしめします（Pythonなどと同様）

no

Route[Idx-1].駅==  
a69list.駅1

no : a)

外外

yes : b)内外

Route[Idx]に、（Route[Idx]路線, a69list駅1）を挿入

yes : c)内内

Route[Idx-1].駅==a69list.駅1

no : d)外内

Route[Idx]路線←t\_rule69路線

Route[Idx]に、（t\_rule69路線, a69list駅2）を挿入

Route[Idx]に、（Route[Idx]路線, a69list駅1）を挿入

Idx-2からRoute重複路線削除

置換例2  
(Not 置換例1)

#### 置換例3-1 勝浦→東京

a69list

|  |  |  |
| --- | --- | --- |
| 京葉 | **蘇我** | **東京** |

t\_rule69 recordset SQL

|  |  |  |
| --- | --- | --- |
| 総武 | **東京** | 千葉 |
| 外房 | 千葉 | **蘇我** |

a), b) 非該当

c) 内→内

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 蘇我 | ⇒ |  |  | 蘇我 |  |
| → | 京葉 | 東京 |  | **外房線** | **千葉** |
|  |  |  | → | **総武線** | 東京 |

d) 外→内

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *勝浦* | ⇒ |  |  | *勝浦* | ⇒ |  | *勝浦* |
|  | 外房線 | 蘇我 |  | **外房線** | 蘇我 | 外房線 | 千葉 |
| → | 京葉線 | 東京 |  | **外房線** | **千葉** | 総武線 | 東京 |
|  |  |  | → | **総武線** | 東京 |  |  |

#### 置換例3-2東京→勝浦

a69list

|  |  |  |
| --- | --- | --- |
| 京葉 | **東京** | **蘇我** |

t\_rule69 recordset SQL

|  |  |  |
| --- | --- | --- |
| 総武 | 東京 | 千葉 |
| 外房 | 千葉 | 蘇我 |

a) 非該当

b) 内→外

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 東京 | ⇒ |  |  | **東京** | ⇒ |  | 東京 |
| → | 京葉 | 蘇我 |  | **総武線** | **千葉** | 総武線 | 千葉 |
|  | 外房線 | *勝浦* | → | **外房線** | 蘇我 | 外房線 | *勝浦* |
|  |  |  |  | 外房線 | *勝浦* |  |  |

c) 内→内

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 東京 | ⇒ |  |  | **東京** |  |
| → | 京葉 | 蘇我 |  | **総武線** | **千葉** |
|  |  |  | → | **外房線** | 蘇我 |

d) 外→内※ 外は無視できる（関係ない）例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *川崎* |  |  |  | *川崎* |  |
|  | *東海道* | 東京 | ⇒ |  | *東海道* | **東京** |  |
| → | 京葉 | 蘇我 |  | **総武線** | **千葉** |
|  |  |  | → | **外房線** | 蘇我 |

#### 置換処理(2/3) ：乗車経路(1線)計算経路(2線)に限る実装(a69list is 1 / t\_rule69 is 2)

if a69list駅1==t\_rule69[0]駅1

下り

else

上り

endif

if 下り

Route[Idx]線←t\_rule69[1]線

Insert to (t\_rule69[0]線, t\_rule69[0].駅2)at Route[Idx]

else

Route[Idx]線←t\_rule69[0]線

Insert to (t\_rule69[1]線, t\_rule69[1].駅1)at Route[Idx]

endif

～重複路線を削除（Idx-2～）

#### 置換例4-1 大阪→直江津

a69list

|  |  |  |
| --- | --- | --- |
| 東海道線 | 山科 | 米原 |
| 北陸線 | 米原 | 近江塩津 |

t\_rule69 recordset SQL

|  |  |  |
| --- | --- | --- |
| 湖西線 | 山科 | 近江塩津 |

a) 外→外

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 大阪 | ⇒ |  |  | 大阪 |  |
|  | 東海道線 | 米原 |  | 東海道線 | **山科** |  |
| → | 北陸線 | 直江津 |  | **湖西線** | **近江塩津** |  |
|  |  |  | → | 北陸線 | 直江津 |  |

b) 内→外

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 山科 | ⇒ |  |  | 山科 |  |
|  | 東海道線 | 米原 |  | **湖西線** | **近江塩津** |  |
| → | 北陸線 | 直江津 | → | 北陸線 | 直江津 |  |

c) 内→内

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 山科 | ⇒ |  |  | 山科 |  |
|  | 東海道線 | 米原 | → | **湖西線** | 近江塩津 |  |
| → | 北陸線 | 近江塩津 |  |  |  |  |

d) 外→内

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 大阪 | ⇒ |  |  | 大阪 |  |
|  | 東海道線 | 米原 |  | 東海道線 | **山科** |  |
| → | 北陸線 | 近江塩津 | → | **湖西線** | **近江塩津** |  |

#### 置換例4-2 直江津→大阪

a69list

|  |  |  |
| --- | --- | --- |
| 北陸線 | 近江塩津 | 米原 |
| 東海道線 | 米原 | 山科 |

t\_rule69 recordset SQL

|  |  |  |
| --- | --- | --- |
| 湖西線 | 山科 | 近江塩津 |

a) 外→外

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 直江津 | ⇒ |  |  | 直江津 |  |
|  | 北陸線 | 米原 |  | 北陸線 | **近江塩津** |  |
| → | 東海道線 | 大阪 |  | **湖西線** | **山科** |  |
|  |  |  | → | 東海道線 | 大阪 |  |

追加挿入（米原!=近江塩津なので、北陸線、米原→近江塩津へ置換）、湖西線、山科を追加

b) 内→外

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 近江塩津 | ⇒ |  |  | 近江塩津 |  |
|  | 北陸線 | 米原 |  | **湖西線** | **山科** |  |
| → | 東海道線 | 大阪 | → | 東海道線 | 大阪 |  |

c) 内→内

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 近江塩津 | ⇒ |  |  | 近江塩津 |  |
|  | 北陸線 | 米原 | → | **湖西線** | 山科 |  |
| → | 東海道線 | 山科 |  |  |  |  |

d) 外→内

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 直江津 | ⇒ |  |  | 直江津 |  |
|  | 北陸線 | 米原 |  | 北陸線 | **近江塩津** |  |
| → | 東海道線 | 山科 | → | **湖西線** | **山科** |  |

* 米原までの場合、成立しない（HCPチャートの2段目のループで抜ける）

#### 置換処理(3/3) ：乗車経路(N線)計算経路(1線)の実装(a69list is N / t\_rule69 is 1)

Route[Idx].駅==a69list[-1].駅2

yes : c) or d)

if 2<N then

Route[Idx-(N-2)]～[N-2]行削除

Route[Idx-1]駅←a69list[0]駅1

* N=a69list配列数
* 配列インデックスの’-1’はラストインデックスをしめします（Pythonなどと同様）

no

Route[Idx-N].駅==  
a69list[0].駅1

no : a)

外外

yes : b)内外

Route[Idx-1]路線←t\_rule69路線

Route[Idx-1]駅←a69list[-1]駅2

if 2<N then

Route[Idx-(N-1)]～(N-2)行削除

Route[Idx]に、（t\_rule69路線, a69list[-1]駅2）を挿入

Route[Idx-N]駅←a69list[0]駅1

if 2<N then Route[Idx-(N-1)]～(N-2)行削除

yes : c)内内

Route[Idx-N].駅==a69list[0].駅1

no : d)外内

Route[Idx-(N-1)]～(N-1)行削除

Route[Idx]路線←t\_rule69路線

Route[Idx]駅←a69list[-1]駅2

ア

イ

A

B

C

D

ウ

エ

オ

カ

a69list

|  |  |  |
| --- | --- | --- |
| A | イ | ウ |
| B | ウ | エ |
| C | エ | オ |

t\_rule69

|  |  |  |
| --- | --- | --- |
| D | イ | オ |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | ア |  |  | ア |  |
|  | A | ウ | A | **イ** |  |
|  | B | エ | **D** | **オ** |  |
| → | C | カ | **C** | **カ** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | ア |  |  | ア |  |
|  | A | ウ | A | **イ** |  |
|  | B | エ | **~~B~~** | **~~エ~~** |  |
| → | C | カ | **D** | **オ** |  |
|  |  |  | → | **C** | **カ** |  |

ア

イ

A

B

C

E

ウ

エ

オ

カ

D

キ

a69list

|  |  |  |
| --- | --- | --- |
| A | イ | ウ |
| B | ウ | エ |
| C | エ | オ |
| D | オ | カ |

t\_rule69

|  |  |  |
| --- | --- | --- |
| E | カ | イ |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | ア |  |  | ア |  |
|  | A | ウ | A | **イ** |  |
|  | B | エ | **E** | **カ** |  |
| → | C | オ | **D** | **キ** |  |
|  | D | キ |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | ア |  |  | ア |  |
|  | A | ウ | A | **イ** |  |
|  | B | エ | ~~B~~ | ~~エ~~ |  |
|  | C | オ | ~~C~~ | ~~オ~~ |  |
| → | D | キ |  | **E** | **カ** |  |
|  |  |  | → | D | キ |  |

#### 重複経路除去アルゴリズム（文字列の実装例）

#include <stdio.h>

#include <string.h>

void removeAtIndex(char\* s, int pos)

{

int len = strlen(s);

if (pos < len) {

strcpy(s + pos, s + pos + 1);

}

}

void dupremov(char\* s)

{

int last = strlen(s) - 1;

int idx = 0;

while (idx < last) {

if (s[idx] == s[idx + 1]) {

removeAtIndex(s, idx);

last--;

} else {

idx++;

}

}

}

int main(int argc, char\*\* argv)

{

char buf[128];

strcpy(buf, \*++argv);

dupremov(buf);

printf("%s -> %s\n", buf, \*argv);

return 0;

}

# ■ 特定都市内駅の発着86,87条

特定都市内駅の発着時、中心駅と定められた駅から（まで）200km以上は、中心駅から（まで）計算する。

ただし、特定都区市内から目的駅までの運賃が中心駅から200kmに達する最初の駅までの運賃と、実際に乗車する発駅〜目的駅が200kmに満たない場合で、特定都区市内中心駅から目的駅までの運賃計算キロが大きい場合，前者の運賃を採用する（基114）。すなわち、例として、運賃計算キロが横浜市内の中心駅である横浜駅から200kmに達した最初の駅Aまでの運賃と、横浜駅から200kmに達しない駅で、本郷台からの運賃計算キロが横浜−駅Aまでの運賃を越える場合、駅Aの運賃を採用する。

また、特定都区市内駅からいったんはずれて、再度、発駅（または着駅）と同一の特定都区市内駅を通過する場合は適用しないことができる（適用しない）。

200km／100km以上は、営業キロである。（換算キロではない）

## 実装概要

**~~規則69条の後におこなってから、本適用をおこない、さらに規則69条の適用をおこなう必要がある。~~**

**~~（埼京線－浮島船戸、北赤羽着発）~~**

**本適用は、経路指定時、自動計算はしない。「都区市内発着」ボタン\*1をクリックすることにより、適用する。再度「都区市内発着」ボタンをクリックすると**

**\*1** ・・・「北九州市内発」／「単駅発」、「北九州市内着」／「単駅着」のチェックボタンが選択可能になる。

**本適用後、69条を適用し、201km判定をおこない、未適用の場合、さらに規則87条で再スキャンして100㎞判定をおこなう。**

200㎞／100㎞判定は、86条、69条適用後におこなう

## データ定義

|  |  |  |
| --- | --- | --- |
| テーブルt\_station  city\_flg | | |
| BCCYAMETE | 山手線内駅 | 1ビット |
| BCCITYCT | 中心駅 | 1ビット |
| BCCITYNO | 都区市内番号 | 4ビット |

テーブル：rule86は脱出（進入）路線への経路が中心駅からその路線への乗換え駅と路線を定義する。以下の例1は、常磐線は、東京－東北線－日暮里－常磐線であり、例2は、東京－東北線であることをしめす。

一部大阪市内では例外があって2線以内に到達しない例があり、2レコードにまたがる例がある（「おおさか東線」の例）。この場合、2レコード目のcity\_idのbit8から上位にレコード番号(2レコード目=1)を表すため以下のクエリーとなる。

select \* from t\_rule86 where (city\_id & 255)=?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| テーブルt\_rule86 | | | | |
| 列名 | line\_id1 | station\_id | line\_id2 | city\_id |
| 説明 | 脱出（進入）路線 | 最初の乗換駅 | 最初の路線 | 都区市内番号 |
| 例1  （2線の例） | 常磐線 | 日暮里 | 東北線 | 1 |
| 例2  （1線の例） | 東北 | 東京 | 東北線 | 1 |

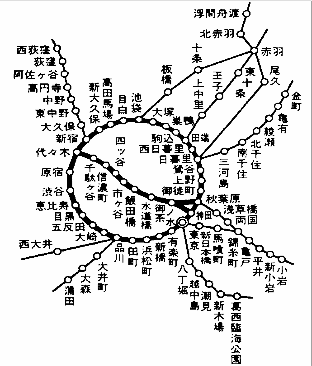
#### 86条／87条適用パターン

|  |  |  |
| --- | --- | --- |
|  | 適用パターン | |
| A |  | 適用 |
| B |  | 対象外 |
| **C** |  | 対象外 |
| D |  | 適用 |
| E |  | 適用 |
| F |  | 対象外（○内最短経路） |
| G |  | 対象外 |
| H |  | 対象外 |

## 規則86条

#### 東京都区内

|  |  |
| --- | --- |
| 東京都区内（東京） | |
| 東京 | 長野・北陸新幹線 |
| 東京 | 上越新幹線 |
| 東京 | 東北新幹線 |
| 品川 | 東海道新幹線 |
| 西荻窪 | 中央東線 |
| 西大井 | 東海道線（品川‐鶴見） |
| 蒲田 | 東海道線 |
| 葛西臨海公園 | 京葉線 |
| 小岩 | 総武線 |
| 金町 | 常磐線 |
| 浮間舟渡 | 東北線(赤羽-大宮)(埼京線) |
| 赤羽 | 東北線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id\_1 | station\_id | line\_id2 | city\_id |
| 常磐線 | 日暮里 | 東北線 | 1 |
| 中央東線 | 東京 | 中央東線 | 1 |
| 東海道線（西大井経由） | 品川 | 東海道線 | 1 |
| 東海道線 | 東京 | 東海道線 | 1 |
| 京葉線 | 東京 | 京葉線 | 1 |
| 総武線 | 東京 | 総武線 | 1 |
| 東北線(埼京線) | 赤羽 | 東北線 | 1 |
| 東北線 | 東京 | 東北線 | 1 |
| 長野・北陸新幹線 | 東京 | 長野・北陸新幹線 | 1 |
| 上越新幹線 | 東京 | 上越新幹線 | 1 |
| 東北新幹線 | 東京 | 東北新幹線 | 1 |
| 東海道新幹線 | 東京 | 東海道新幹線 | 1 |

以降の凡例

|  |  |
| --- | --- |
| **用語** | **DBフィールド名** |
| 脱出路線／進入路線 | line\_id1 |
| 乗換駅 | station\_id |
| 中心駅発着路線 | line\_id2 |

#### ＜パターン：脱出≠＞

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 指定経路 | |  | 規則86条適用 | |  | 規則69条適用 | |
|  | 蒲田 |  |  | **都区内（東京）** |  |  | 都区内（東京） |
| 東海道線 | 品川 |  | **東北線** | **赤羽** |  | 東北線 | 大宮 |
| 山手線 | 代々木 |  | 埼京線 | **大宮** |  | 高崎線 | 高崎 |
| 中央東線 | 御茶ノ水 |  | 高崎線 | 高崎 |  | 上越線 | 宮内 |
| 総武線 | 秋葉原 |  | 上越線 | 宮内 |  | 信越線 | 新津 |
| 東北線 | 田端 |  | 信越線 | 新津 |  | 羽越線 | 秋田 |
| 山手線 | 池袋 |  | 羽越線 | 秋田 |  |  |  |
| 赤羽線 | 赤羽 |  |  |  |  |  |  |
| **埼京線** | **大宮** |  |  |  |  |  |  |
| 高崎線 | 高崎 |  |  |  |  |  |  |
| 上越線 | 宮内 |  |  |  |  |  |  |
| 信越線 | 新津 |  |  |  |  |  |  |
| 羽越線 | 秋田 |  |  |  |  |  |  |

脱出路線（埼京線）≠中心駅発着路線（東北線）なので脱出路線の行の前に行挿入しその行を中心駅発着路線（東北線）と乗換駅（赤羽）とし、その前の行の駅を中心駅（東京）（flg.BCCITYNO=cityflg.BCCITYNOの都区市内番号を設定）にする。それ以前の行の路線は削除する。（2段目列）

その後、規則69条適用（3段目列）

#### ＜パターン：進入≠＞

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 指定経路 | |  | 規則86条適用後経路 | |  | 規則69条適用 | |
|  | 秋田 |  |  | 秋田 |  |  | 秋田 |
| 羽越線 | 新津 |  | 羽越線 | 新津 |  | 羽越線 | 新津 |
| 信越線 | 宮内 |  | 信越線 | 宮内 |  | 信越線 | 宮内 |
| 上越線 | 高崎 |  | 上越線 | 高崎 |  | 上越線 | 高崎 |
| 高崎線 | 大宮 |  | 高崎線 | 大宮 |  | 高崎線 | 大宮 |
| **埼京線** | **赤羽** |  | 埼京線 | **赤羽** |  | 東北線 | 東京 |
| 赤羽線 | 池袋 |  | **東北線** | **東京** |  |  |  |
| 山手線 | 田端 |  |  |  |  |  |  |
| 東北線 | 秋葉原 |  |  |  |  |  |  |
| 総武線 | 御茶ノ水 |  |  |  |  |  |  |
| 中央東線 | 代々木 |  |  |  |  |  |  |
| 山手線 | 品川 |  |  |  |  |  |  |
| 東海道線 | 蒲田 |  |  |  |  |  |  |

進入路線（埼京線）≠中心駅発着路線（東北線）なので、その直後を中心駅発着路線（東北線）と中心駅（東京）（東京都区内）とする（flg.BCCITYNO=cityflg.BCCITYNOの都区市内番号を設定）。進入路線の駅は、乗換駅（赤羽）に置換。進入路線の行+2以降の行は削除する。（2段目列）

その後、規則69条適用（3段目列）

#### ＜パターン：脱出＝＞

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 蒲田 |  |  | 東京 | flg.BCCITYNO=東京都区内 |
| 東海道線 | 品川 |  | 東北線 | 盛岡 |  |
| 山手線 | 代々木 |  |  |  |  |
| 中央東線 | 御茶ノ水 |  |  |  |  |
| 総武線 | 秋葉原 |  |  |  |  |
| **東北線** | **盛岡** |  |  |  |  |

脱出路線（東北線）＝中心駅発着路線（東北線）なので脱出路線の行-1の行の駅を中心駅（東京）（flg.BCCITYNO=cityflg.BCCITYNOの都区市内番号を設定）にし、路線は削除。それ以前（脱出路線の行-2以前）の行は削除する。（2段目列）

その後、規則69条適用（3段目列）

#### ＜パターン：進入＝＞

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 盛岡 |  |  | 盛岡 |  |
| **東北線** | **秋葉原** |  | 東北線 | 東京 | flg.BCCITYNO=東京都区内 |
| 総武線 | 御茶ノ水 |  |  |  |  |
| 中央東線 | 代々木 |  |  |  |  |
| 山手線 | 品川 |  |  |  |  |
| 東海道線 | 蒲田 |  |  |  |  |

進入路線（東北線）＝中心駅発着路線（東北線）なので、進入路線の行の駅を中心駅（東京駅）（東京都区内）とする（flg.BCCITYNO=cityflg.BCCITYNOの都区市内番号を設定）。それ以降（進入路線の行+1以降）の行は削除する。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 蒲田 |  |  | 東京 |
| 東海道線 | 東京 |  | 東北線 | 盛岡 |
| **東北線** | **盛岡** |  |  |  |

（同上）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 赤羽 |  |  | 東京 |
| **東北線** | **盛岡** |  | 東北線 | 盛岡 |

（同上）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 指定経路 | |  | 規則86条適用後経路 | |  | 規則69条適用 | |
|  | 赤羽 |  |  | **東京** |  |  | 東京 |
| **埼京線** | **大宮** |  | **東北線** | **赤羽** |  | 東北線 | 盛岡 |
| 東北線 | 盛岡 |  | 埼京線 | 大宮 |  |  |  |
|  |  |  | 東北線 | 盛岡 |  |  |  |

＜パターン：脱出≠＞

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 蒲田 |  |  | 東京 |
| 東海道線 | 東京 |  | 中央東線 | 西国分寺 |
| **中央東線** | **西国分寺** |  | 武蔵野線 | 武蔵浦和 |
| 武蔵野線 | 武蔵浦和 |  | 埼京線 | 大宮 |
| 埼京線 | 大宮 |  | 東北線 | 盛岡 |
| 東北線 | 盛岡 |  |  |  |

＜パターン：脱出＝＞

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 浮間舟渡 |  |  | 東京 |
| 埼京線 | 赤羽 |  | 中央東線 | 松本 |
| 赤羽線 | 池袋 |  |  |  |
| 山手線 | 新宿 |  |  |  |
| **中央東線** | **松本** |  |  |  |

＜パターン：脱出＝＞

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指定経路 | |  | 86条適用 | |  | 69条適用 | |  | 重複最適化 | |
|  | 浮間舟渡 |  |  | 東京 |  |  | 東京 |  |  | 東京 |
| **埼京線** | **大宮** |  | **東北線** | **赤羽** |  | 東北線 | 赤羽 |  | 東北線 | 盛岡 |
| 東北線 | 盛岡 |  | 埼京線 | 大宮 |  | 東北線 | 盛岡 |  |  |  |
|  |  |  | 東北線 | 盛岡 |  |  |  |  |  |  |

＜パターン：脱出≠＞

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指定経路 | |  | 86条適用 | |  | 69条適用 | |  | 重複最適化 | |
|  | 盛岡 |  |  | 盛岡 |  |  | 盛岡 |  |  | 盛岡 |
| 東北線 | 大宮 |  | 東北線 | 大宮 |  | 東北線 | 赤羽 |  | 東北線 | 東京 |
| **埼京線** | **浮間舟渡** |  | 埼京線 | **赤羽** |  | 東北線 | 東京 |  |  |  |
|  |  |  | **東北線** | 東京 |  |  |  |  |  |  |

＜パターン：進入≠＞

（2段目列）

その後、規則69条適用（3段目列）

69条適用(3段目列)

重複路線削除(4段列目)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 浮間舟渡 |  |  | 東京 |
| 埼京線 | 赤羽 |  | 東北線 | 盛岡 |
| **東北線** | **盛岡** |  |  |  |

＜パターン：脱出＝＞

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 浮間舟渡 |  |  | **都区内** |  |
| **埼京線** | **武蔵浦和** |  | **東北線** | **赤羽** |  |
| 武蔵野線 | 南浦和 |  | 埼京線 | 武蔵浦和 |  |
| 東北線 | 盛岡 |  | 武蔵野線 | 南浦和 |  |
|  |  |  | 東北線 | 盛岡 |  |

＜パターン：脱出≠＞

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 浮間舟渡 |  |  | **東京** |  |
| **埼京線** | **武蔵浦和** |  | **東北線** | **赤羽** |  |
| 武蔵野線 | 新松戸 |  | 埼京線 | 武蔵浦和 |  |
| 常磐線 | 岩沼 |  | 武蔵野線 | 新松戸 |  |
| 東北線 | 盛岡 |  | 常磐線 | 岩沼 |  |
|  |  |  | 東北線 | 盛岡 |  |

＜パターン：脱出≠＞

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 板橋 |  |  | 東京 |
| 赤羽線（埼京線） | 赤羽 |  | 東海道新幹線 | 名古屋 |
| 東北線（尾久経由） | 日暮里 |  |  |  |
| 東北線 | 東京 |  |  |  |
| **東海道新幹線** | **名古屋** |  |  |  |

＜パターン：脱出＝＞

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 板橋 |  |  | 東京 |
| 赤羽線（埼京線） | 赤羽 |  | 東海道新幹線 | 名古屋 |
| 東北線（尾久経由） | 日暮里 |  |  |  |
| 東北線 | 秋葉原 |  |  |  |
| 総武線（御茶ノ水-錦糸町） | 御茶ノ水 |  |  |  |
| 中央東線 | 代々木 |  |  |  |
| 山手線 | 品川 |  |  |  |
| **東海道新幹線** | **名古屋** |  |  |  |

＜パターン：脱出＝＞

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 蒲田 |  |  | 東京 |
| **東海道線** | **川崎** |  | 東海道線 | 川崎 |
| 南武線 | 立川 |  | 南武線 | 立川 |
| 中央東線 | 松本 |  | 中央東線 | 松本 |

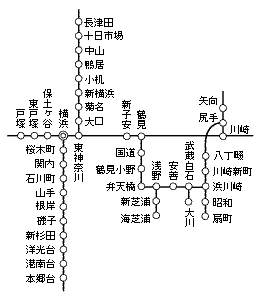
＜パターン：脱出＝＞

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | 東京 |
|  | 西大井 |  | 東海道線 | 品川 |
| **東海道線(西大井経由)** | **武蔵小杉** |  | 東海道線（西大井経由） | 武蔵小杉 |
| 南武線 | 府中本町 |  | 南武線 | 府中本町 |
| 武蔵野線 | 新松戸 |  | 武蔵野線 | 新松戸 |
| 常磐線 | 岩沼 |  | 常磐線 | 岩沼 |

＜パターン：脱出≠＞

#### 横浜市内

|  |  |
| --- | --- |
| 横浜市内（横浜） | |
| 長津田 | 横浜線 |
| 戸塚 | 東海道線 |
| 本郷台 | 根岸線 |
| 川崎 | 東海道線 |
| 矢向 | 南武線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 横浜線 | 東神奈川 | 東海道線 | 2 |
| 東海道線 | 横浜 | 東海道線 | 2 |
| 根岸線 | 横浜 | 根岸線 | 2 |
| 南武線 | 川崎 | 東海道線 | 2 |
| 東海道新幹線 | 新横浜 | 東海道新幹線 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 海尻 |  |  |  |
| 小海線 | 小淵沢 |  |  |  |
| 中央東線 | 八王子 |  |  |  |
| 横浜線 | 長津田 |  |  |  |

#### 名古屋市内

|  |  |
| --- | --- |
| 名古屋市内（名古屋） | |
| 名古屋 | 東海道線、東海道新幹線 |
| 春田 | 関西線 |
| 南大高 | 東海道線 |
| 新守山 | 中央西線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 東海道新幹線 | 名古屋 | 東海道新幹線 | 3 |
| 東海道線 | 名古屋 | 東海道線 | 3 |
| 中央西線 | 金山 | 東海道線 | 3 |
| 関西線 | 名古屋 | 関西線 | 3 |

大高―（東海道線、大阪環状線）－杉本町は、

大阪―名古屋が、190.4kmであるが、杉本町―名古屋208.0km、大阪―大高202.8kmであるので、大阪市内―大高または、杉本町―名古屋市内のいずれかでOK.

ただし、デフォルトでは申し出のない場合、着駅を特定都市内で発券される。

ただし、名古屋―杉本町と指定した場合、名古屋市内―杉本町となる

杉本町 ― 大阪

名古屋　―　笠寺　―　 大高

大阪市内

名古屋市内

202.8km

208.0km

199.6km

190.4km

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 発 | 着 | 発券 |  |  |
| 大高 | 杉本町 | 大高―大阪市内 | 202.8km | 名古屋―大阪190.4kmで満たないので  名古屋―杉本町なら満たす  大高―大阪市内も満たすので着駅優先で後者採用 |
| 笠寺 | 杉本町 | 名古屋市内―杉本町 | 208.0km | 名古屋―大阪190.4kmで満たない  笠寺―大阪199.6kmで満たないので、  名古屋市内―杉本町となる |

A s---c-------c----g

B c---s-------c----g

C s---c-------g----c

D c---s-------g----c

Start, Center, Goal

if c - c > 200

c - c

else if s - c > 200

s - c //A

else if c - g > 200

c - g //D

else

#### 京都市内

|  |  |
| --- | --- |
| 京都市内（京都） | |
| 保津峡 | 山陰線 |
| 桂川 | 東海道線 |
| 桃山 | 奈良線 |
| 山科 | 東海道線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 東海道新幹線 | 京都 | 東海道新幹線 | 4 |
| 東海道線 | 京都 | 東海道線 | 4 |
| 山陰線 | 京都 | 山陰線 | 4 |
| 奈良線 | 京都 | 奈良線 | 4 |
| 湖西線 | 山科 | 東海道線 | 4 |

#### 大阪市内

|  |  |
| --- | --- |
| 大阪市内（大阪） | |
| 塚本 | 東海道線 |
| 加島 | JR東西線 |
| 杉本町 | 阪和線 |
| 加美 | 関西線 |
| 放出 | 片町線 |
| 東淀川 | 東海道線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 東海道新幹線 | 新大阪 | 東海道新幹線 | 5 |
| 東海道線 | 大阪 | 東海道線 | 5 |
| 山陽新幹線 | 新大阪 | 山陽新幹線 | 5 |
| おおさか東線 | 放出 | 片町線 | 5 |
| おおさか東線 | 京橋 | 大阪環状線 | 261(0x105) |
| JR東西線 | 京橋 | 大阪環状線 | 5 |
| 片町線 | 京橋 | 大阪環状線 | 5 |
| 関西線 | 天王寺 | 大阪環状線 | 5 |
| 阪和線 | 天王寺 | 大阪環状線 | 5 |

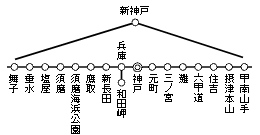
* おおさか東線は2レコード定義で、city\_id>0xffは2レコード目を指す。  
  したがって、脱出路線が「おおさか東線」の場合、以下となる
* （脱出のパターン：＝はあり得ない）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 放出 |  |  | **大阪** |
| **おおさか東線** | **久宝寺** | **大阪環状線** | **京橋** |
| 関西線 | 王子 | **片町線** | **放出** |
| 和歌山線 | 和歌山 | おおさか東線 | 久宝寺 |
| 紀勢線 | 亀山 | 関西線 | 和歌山 |
| 関西線 | 名古屋 | 紀勢線 | 亀山 |
|  | | 関西線 | 名古屋 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 名古屋 | |  |  | 名古屋 |
| 関西線 | 亀山 | | 関西線 | 亀山 |
| 紀勢線 | 和歌山 | | 紀勢線 | 和歌山 |
| 和歌山線 | 王子 | | 和歌山線 | 王子 |
| 関西線 | 久宝寺 | | 関西線 | 久宝寺 |
| **おおさか東線** | | **放出** | おおさか東線 | **放出** |
|  | | | **片町線** | **京橋** |
| **大阪環状線** | **大阪** |

#### 神戸市内

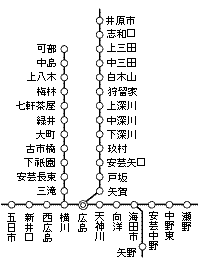
|  |  |
| --- | --- |
| 神戸市内（神戸） | |
| 舞子 | 山陽線 |
| 甲南山手 | 東海道線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 山陽線 | 神戸 | 山陽線 | 6 |
| 山陽新幹線 | 新神戸 | 山陽新幹線 | 6 |
| 東海道線 | 神戸 | 東海道線 | 6 |

#### 広島市内

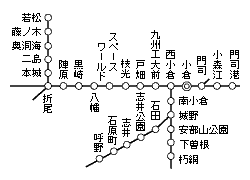
|  |  |
| --- | --- |
| 広島市内（広島） | |
| 井原市 | 芸備線 |
| 五日市 | 山陽線 |
| 矢野 | 呉線 |
| 瀬野 | 山陽線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 山陽線 | 広島 | 山陽線 | 7 |
| 山陽新幹線 | 広島 | 山陽新幹線 | 7 |
| 芸備線 | 広島 | 芸備線 | 7 |
| 可部線 | 横川 | 山陽線 | 7 |
| 呉線 | 海田市 | 山陽線 | 7 |

#### 北九州市内

|  |  |
| --- | --- |
| 北九州市内（小倉） | |
| 折尾 | 筑豊線、鹿児島線 |
| 呼野 | 日田彦山線 |
| 朽網 | 日豊線 |
| 門司 | 鹿児島線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 鹿児島線 | 小倉 | 鹿児島線 | 8 |
| 日豊線 | 小倉 | 日豊線 | 8 |
| 日田彦山線 | 城野 | 日豊線 | 8 |
| 筑豊線 | 折尾 | 鹿児島線 | 8 |

#### 福岡市内

|  |  |
| --- | --- |
| 福岡市内（博多） | |
| 博多 | 博多南線 |
| 博多 | 九州新幹線 |
| 博多 | 山陽新幹線 |
| 南福岡 | 鹿児島線 |
| 土井 | 香椎線 |
| 福工大前 | 鹿児島線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 鹿児島線 | 博多 | 鹿児島線 | 9 |
| 篠栗線 | 吉塚 | 鹿児島線 | 9 |
| 香椎線 | 香椎 | 鹿児島線 | 9 |
| 九州新幹線 | 博多 | 九州新幹線 | 9 |
| 山陽新幹線 | 博多 | 山陽新幹線 | 9 |

#### 仙台市内

|  |  |
| --- | --- |
| 仙台市内（仙台） | |
| 奥新川 | 仙山線 |
| 南仙台 | 東北線 |
| 中野栄 | 仙石線 |
| 岩切 | 東北線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 東北線 | 仙台 | 東北線 | 10 |
| 東北新幹線 | 仙台 | 東北新幹線 | 10 |
| 仙石線 | 仙台 | 仙石線 | 10 |
| 仙山線 | 仙台 | 仙山線 | 10 |

#### 札幌市内

|  |  |
| --- | --- |
| 札幌市内（札幌） | |
| あいの里公園 | 札沼線 |
| ほしみ | 函館線 |
| 森林公園 | 函館線 |
| 上野幌 | 千歳線 |



|  |  |  |  |
| --- | --- | --- | --- |
| t\_rurle86 | | | |
| line\_id1 | station\_id | line\_id2 | city\_id |
| 函館線 | 札幌 | 函館線 | 11 |
| 札沼線 | 桑園 | 函館線 | 11 |
| 千歳線 | 白石 | 函館線 | 11 |

## 実装詳細

本適用後、69条を適用後、201km判定をおこない、未適用の場合、さらに規則87条で再スキャンして100㎞判定をおこなう。

200㎞／100㎞判定は、86条、69条適用後におこなう

#### 86／87条判定フロー

86条発着チェック

87条発着チェック

下表の判定

営業キロ←200km

営業キロ←100km

1,2,3,0x83

1,2,3,0x83以外

1,2,0x83

1,2,3,0x83以外

置換処理

100 or 200km以上

100 or 200km未満

下表の200km判定

200km未満

1. or(2)を通った？

or 1 or 2?

(1)

(2)

114条チェック

Yes

No

86条発着チェック

87条発着チェック

営業キロ←200km

営業キロ←100km

1,2,3,0x83

1,2,3,0x83以外

1,2,3, 0x83

1,2,3,0x83以外

（0 or 4）

置換処理

200km以上

100 or 200km未満

下表の200km判定

200km未満

114条チェック

下表の判定

100 km以上

都区市内発着でないまたは、都区市内完結

都区市内発着

|  |  |  |
| --- | --- | --- |
| 戻値 | 発着チェック結果 | 判定処理 |
| 0 | 発駅、着駅とも単駅（通常駅） | ― |
| 1 | 発駅のみ特定都市内駅 | 発駅を特定都区市内に一時的に置き換えて営業キロが満たしているか判定 |
| 2 | 着駅のみ特定都市内駅 | 着駅を特定都区市内に一時的に置き換えて営業キロが満たしているか判定 |
| 3 | 発駅、着駅とも特定都市内駅 | 発着駅を特定都区市内に一時的に置き換えて営業キロが満たしているか判定 |
| 0x83 | 発駅、着駅とも同一特定都市内駅 |
| 4 | 全駅同一特定都市内駅 | ― |

* 200km越えていれば86条なので、86条駅⇔87条駅はありえない。

#### 特定都市内駅の発着（規則86、87条の適用）チェック

**86条チェック**

int checkOfSpecificStartAndEndCityTerminal(route)

route : RouteItem配列

return 0: 発駅、着駅とも単駅（通常駅）

1: 発駅のみ特定都市内駅

2: 着駅のみ特定都市内駅

3: 発駅、着駅ともそれぞれ別々の特定都市内駅

0x83： 発駅、着駅とも同一特定都市区内（Oの字）

4: 全駅同一特定都市内駅

全ルート配列を対象に特定都区市内発着チェックをおこなう（引数はRoute配列）

**87条チェック**

int checkOfSpecificStartAndEndCityTerminalTM(route)

route : RouteItem配列

return 0: 発駅、着駅とも単駅（通常駅）

1: 発駅のみ特定都市内駅

2: 着駅のみ特定都市内駅

3: 発駅、着駅ともそれぞれ別々の特定都市内駅（ありえない）

0x83： 発駅、着駅とも同一特定都市区内（Oの字）

4: 全駅同一特定都市内駅

全ルート配列を対象に東京山手線内発着チェックをおこなう（引数はRoute配列）

TMは、Tokyo Metroの略

#### 置換処理

先述の「＜パターン：脱出≠＞」、「＜パターン：進入≠＞」、「＜パターン：脱出＝＞」、「＜パターン：進入＝＞」（59～60ページ）を参照

次ページに86条チェックのフローをしめす。

87条チェックは、86条チェックの処理と同等で簡素化されたものとなる。

発駅== 特定都区市内→cityno

c = 0

route配列（発駅～着駅）→ite

c == 0

! 発駅特定都区市内

c++ // 1

発駅特定都区市内

c == 1

assert(c==2)

c++ //2

c ≠ 1

! 発駅特定都区市内

c++ // 3

c == 2

c == 3

c == 2

c == 1

c == 0

return 4

全駅特定都区市内

r = 1

return 0x83

\*1  
c(state no)=

0 そのままo

1 いったん離れたまんま

2 いったん離れて戻ってきたまま(O文字)

3 いったん離れて戻ってきてまた離れた  
(6の字 or 8の字)

規則86条の例外条件により、特定都区市内発着でも発駅は単駅とする（8の字)

規則86条により、発駅から着駅が200㎞を越える場合、発駅を代表駅として計算する(Oの字)

**着駅**== 特定都区市内（発駅の特定都区市内とは異なる都区市内）

c = 0

route配列（**着駅～発駅**）→ite（逆イテレータ）

c == 0

! **着**駅特定都区市内

c++ // 1

**着**駅特定都区市内

c == 1

assert(c==2)

c++ //2

c ≠ 1

c++ // 3

c == 2

c == 3

c == 1

else

assert(false)

r |= 0x02

\*2

c=

0 ありえない（return 4で抜けているから）

1 いったん離れたまんま

2 ありえない

3 いったん離れて戻ってきてまた離れた(6文字)

規則86条の例外条件により、特定都区市内発着でも発駅、着駅とも単駅とする（8の字)

なのでrのビットをたてない

規則86条により、発駅から着駅が200㎞を越える場合、発駅を代表駅として計算する(|の字)

r == 0

return 0

return r

else

c != 0

c != 0

! **着**駅特定都区市内

\*1

\*2

r = 0

r=3 発駅：特定都区市内＋着駅：特定都区市内（発駅都区市内≠着駅都区市内）

r=1 発駅：特定都区市内＋着駅：単駅

r=2 発駅：単駅＋着駅：特定都区市内

checkOfSpecificStartAndEndCityTerminal(route)

図 5

Oの字は同一特定都区市内駅の発着（発駅に戻る）のほかに、発駅も着駅も（発駅と着駅の異なる）特定都区内である場合も含む（都区内-都区内や都区内-札幌市内）。

bool Route::checkOfRuleSpecificCoreLine(int disCity, int\* rule114)

最新(2013-Jan)フロー（ルート：概要フロー）

↑大阪―名古屋間の（あ）の適用可否は？、88条、69条チェックはどのタイミングでいれる？

N: 1,2,3,0x83

No

Yes

Y:1,2,3,0x83以外

（0 or 4）

114条チェック

87条チェック

都区市内発着でないまたは、都区市内完結

発着いずれか都区市内か（全駅同一都区市内を除く）

都区市内発着チェック

（86条チェック）

1,2,3,0x83,0 or 4を返す

都区市内仮適用(発,着の適用可能なもの)

発～着は201km以上？

発～着101km以上？

都区市内仮適用  
(着のみ適用)

Yes

山手線発着適用

発着とも都区市内

Yes：[名]-[阪]、[九]-[福]、[区]-[浜]、Loop含む

No

発～着は201km以上？

着のみ都区市内適用

都区市内仮適用  
(発のみ適用)

発～着は201km以上？

発のみ都区市内適用

発着とも単駅  
（変換なし）

発・着を都区市内適用（適用可能なもの）

No

Yes

No：発、着いずれかのみ都区市内

Yes&着駅=単駅指定ではない

No＋着=単駅指定

No＋発=単駅指定

Yes

発～着はsk km以上？

No

sk=190㎞

sk=90㎞

100km以下

69条も適用

69条も適用

69条も適用

200km以下

[山][区]:101～200km

else:100km以下

87有効かつsk2は201km

[札]－[区]

青森‐[区]

[九]‐岡山など

Yes&発駅=単駅指定ではない

sk2=201km

sk2=101km

No

Yes

※ 87有効はsk=90

87有効&sk km=190&発～着は90~100km?

sk km=90

Yes

No

単駅指定の場合の扱い

距離を算定するため、86条適用前に69変換する。86適用後は不要である。

86通過パターンは？？？

70条を一番先に変換

69条は一番先に変換しておく。

その後、86をやる

69条→70条→86条？

70条→69条→86条？　←

杉本町 大阪 名古屋 大高

~~発駅、着駅とも都区市内で、発着200km以下の時~~

|  |  |  |  |
| --- | --- | --- | --- |
| ~~発~~ | ~~着~~ | ~~201km以上~~ |  |
| ~~-~~ | ~~-~~ | ~~o~~ |  |
| ~~-~~ | ~~-~~ | ~~-~~ | ~~[区]－[浜]など~~ |
| ~~-~~ | ~~o~~ | ~~o~~ | ~~着を適用~~ |
| ~~-~~ | ~~o~~ | ~~-~~ | ~~発を適用してみる↓~~ |
| ~~o~~ | ~~-~~ | ~~o~~ | ~~発を適用~~ |
| ~~o~~ | ~~-~~ | ~~-~~ | ~~単駅発着~~ |
| ~~o~~ | ~~o~~ | ~~o~~ | ~~[札]－[区]など~~ |
| ~~o~~ | ~~o~~ | ~~-~~ |  |

o適用、-非適用

名古屋、金山、熱田、笠寺、大高

杉本町、天王寺、大阪、東淀川

名古屋―大阪 190.4km

|  |  |  |  |
| --- | --- | --- | --- |
| 指定区間 | MARS.EXE計算変換区間 |  |  |
| 大高―天王寺 | 大高－[阪] | 202.8 | 天王寺を「単駅」にすると、[名]－天王寺201.1  発着とも[単駅]にすると、213.5 |
| 天王寺‐大高 | 天王寺‐[名] | 201.1 | 大高を「単駅」にすると、[阪]－大高 |
| 大高－東淀川 | 大高－[阪] | 202.8 | 東淀川または発着とも「単駅」にすると、198.3 |
| 笠寺―天王寺 | [名]－天王寺 | 201.1 | 笠寺(「発」)のみ、または発着「単駅」にすると210.3　※笠寺-大阪は199.6kmなので[阪]とならない |
| 金山―東淀川 | 金山―東淀川 | 189.2 |  |
| 金山―杉本町 | [名]－杉本町 | 208.0 |  |
| 杉本町‐金山 | 杉本町‐[名] | 208.0 |  |
| 笠寺－杉本町 | [名]－杉本町 | 208.0 | 笠寺－大阪は199.6km |
| 杉本町‐笠寺 | 杉本町‐[名] | 208.0 |  |

着を適用してみて201km以上なら、そのまま単駅－[都区市内]とする（天王寺―大高の例）

着を適用してみて200km以下（笠寺－杉本町や天王寺[大阪]）の場合、発を適用してみて201km以上なら、そのまま[都区市内]－単駅とする（笠寺－杉本町や天王寺）、200km以下なら単駅発着とする。

# ■ 基準規定114条

特定都区市内にある駅と、その代表駅から営業キロ200km以下の区間にある駅との相互間についての鉄道の旅客運賃は、その関連する特定都区市内の代表駅から同一の方向および経路による営業キロが200kmを超える区間にある駅との普通旅客運賃に比較して、これよりも高額となる場合は、その同一の方向および経路による規則86条の規定を受ける最近の駅までの普通旅客運賃をもってこの区間の旅客運賃とすることができる。

（東京山手線発着の規則87条における100kmの規定についても同様）

## データ定義

|  |  |  |
| --- | --- | --- |
| テーブルt\_station  city\_flg | | |
| BCYAMETE | 山手線内駅 | 1ビット |
| BCCITYCT | 中心駅 | 1ビット |
| BCCITYNO | 都区市内番号 | 4ビット |
| ~~BCC114~~ | ~~都区市内中心駅から100km、200kmに達する最初の駅（114条チェック用）(BCCITYNO≠0かつ、spe\_route.BSRC114≠0の路線でのみ有効)~~ | ~~1ビット~~ |

|  |  |  |
| --- | --- | --- |
| テーブルt\_lines  specifit\_flg | | |
| ~~BSRC114~~ | ~~00：無効~~  ~~01：下り線で有効~~  ~~10：上り線で有効~~ | ~~2ビット~~ |
| ~~BSRD114~~ | ~~中心駅からの運賃計算キロ‐100km／200km(100.0/200.0～151.1/251.1km)~~ | ~~5ビット~~ |

＜適用される例＞

長津田から[東神奈川]、東海道線、[富士]、身延線経由

|  |  |  |  |
| --- | --- | --- | --- |
| 長津田 | 横浜 | 国母 | 甲斐住吉 |
| 19.7 | 0  0 | 198.6  206.7 | 200.5  208.8 |
|  |  | \3,570- | |
|  | | 218.3  226.4 |  |
| \3,890 |  |
|  | | | \3,570- |

長津田

単駅

(市)

横浜

国母

甲斐住吉

富士

**\3,570－**

**\3,890－**

**\3,570－**

198.6(206.7)km

200.5(208.8)km

200.5(208.8)km

218.3(226.4)km

長津田－国母 \3,890-(横浜－国母が200km以内なので長津田から計算)

長津田－甲斐住吉 \3,570-（横浜－甲斐住吉が200kmを越えるので、横浜から計算）

上記のとおり国母より遠い甲斐住吉の方が安いので、甲斐住吉までの料金と同一とする。

他に、井原市から広島を経て山陽本線上り方面など

A）　御茶ノ水－岩舟（代々木、原宿、田町、上野、土呂、小山経由）

123.6km／\2,210-

B）　御茶ノ水－岩舟（秋葉原、東北線、両毛線経由：A経路での乗車可能。但し100km未満なので途中下車ダメ）

98.8km／\1,620-

C）　東京―岩舟 99.9km／\1,620-

D）　東京―佐野 107.2km／\1,890-

* 東京から、岩舟の先の隣駅が佐野

A)は、東京(最短距離)岩舟が100km未満なので単駅計算で、\2,210であるが、岩舟の一つ先の佐野は、100kmを越えるので山手線内発で、\1,890-（Dで計算）と安いため、\1,890-として発券可能（マルスは非対応らしい）（新幹線は除外）

B)は、157条2項（近郊区間）による発券で、通常発券される切符。ただし途中下車はできない（新幹線は除外）

C) は100km未満なので適用されない（参考）

D)は、100km以上なので、山手線内着発が適用され、山手線内各駅から佐野までの切符はすべて東京駅発で計算される（新幹線は除外）

――――――――――――――――――――――――――

A)の経路で、御茶ノ水～岩舟を発券

1. 東京から岩船までは100km未満(99.9km：①)なので単駅計算→ \2,210
2. 近郊区間なので157条2項適用で、最短経路で計算→ \1,620
3. 途中下車したいので157条条2項非適用で、→ \2,210
4. 東京‐岩船が99.9kmと微妙なので(114条候補)、岩舟の先の100kmに達する最初の駅（岩舟のさきの佐野）までの運賃を算出→ 107.2km／\1,890－
5. \2,210より安いので、114条適用可能でAの経路は、\1,890で発券可能

＜適用されない例＞

本郷台

単駅

(市)

横浜

上諏訪

下諏訪

八王子

**\3,570－**

**\3,570－**

198.9km

203.3km

217.4km

横浜－（横浜線、[八王子]、中央東線経由）－上諏訪 198.9km \3,260-

横浜－（横浜線、[八王子]、中央東線経由）－下諏訪 203.3km \3,570-

本郷台－（横浜線、[八王子]、中央東線経由）－上諏訪 217.4km \3,570-

営業キロは逆転するが、料金は同一に納まっている。

## 実装詳細

「86／87条判定フロー」（68ページ）の処理内でおこなわれる。

以下の条件に合致したものについて114条かどうかを判定する

* 発着駅は、86条、87条候補だが、営業キロが、200km、100kmに満たない為非該当で、営業キロは180～200.0km or 80～100km

について以下判定処理を実施（それ以外は単駅の運賃計算キロ（下記（A））のままとし114条非適用とする）

判定処理

* 単駅の運賃計算キロを算出 - （A）

(最初の例の「長津田」－「国母」間の運賃計算キロ226.4km)

着駅のみ特定都区市内or発駅のみ特定都区市内

着駅

発駅

他

発駅の路線は下り乗車？

{発駅路線}のspe\_route.BSRC114={**01}**かつcity\_flag.BCC114=1かつcity\_flag.BCCITYNO=特定都区市内駅のBSRD114(中心駅からの運賃計算キロ)を得る（1駅のはず）

都区市内

単駅

都区市内

単駅

Yes

{1} = 着駅路線

{2} = **10**

最初の例の「横浜」－「甲斐住吉」（208.8km）

－(B)

(B) < (A) ？

114条適用し、運賃計算キロを（B）とする

114条非適用とし、運賃計算キロを（A）のままとする

Yes

N: 上り

No

着駅のみ特定都区市内or発駅のみ特定都区市内

着駅が特定都区市内(ア)(ウ)

発駅が特定都区市内(イ)(エ)

Yes(イ)

下り

クエリー

(A)-(B)間86・87条適用時の営業キロ：cs 86・87条判定時

(A)-(B)間86・87条適用時の計算キロ：cc に得られる

(A)-(B)間86・87条非適用時の営業キロ：os 以下フローで

(A)-(B)間86・87条非適用時の営業キロ：oc 取得

(C)-(D)間営業キロ： is114s, 計算キロ：is114c

(b) < (a) ？

114条適用し、運賃計算キロを（b）とする

114条非適用とし、運賃計算キロを（a）のままとする

Yes

No

発駅路線は下り？

長津田

横浜

(中心駅)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 富士 | 国母 | 甲斐住吉 |
| (身延線) | 0 | 812  893 | 834  914 |
| 横浜から | 1174 | 1986  2067 | 2005  2088 |
| 長津田から | 1371 | 2183  2264 | 2202  2285 |

200.0km

197.7km

(ア)

(イ)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 広島 | 和気 | 吉永 | 三石 |
| （神戸から） | 3047 | 1148 | 1095 | 1024 |
|  | 0 | 1899 | 1952 | 2023 |
| 井原市から | 371  408 | 2270  2307 | 2323  2360 | 2394  2431 |

200.0km

371

408

(ウ)

(エ)

井原市

上り←身延線→下り

下り←芸備線→上り

下り←山陽線→上り

No(ア)

上り

最終路線は下り？

No(エ)

上り

Yes(ウ)

下り

クエリー

クエリー

クエリー

(A)

(A)

(B)

(B)

os, ocを取得

ocから運賃を取得→(a)

上記クエリーで、100／200km到達最短営業キロ(is114s)駅を取得、is114cから(C)-(D)間運賃（b）を得る

(C)

(D)

End

Start

井原市－和気 227.0km／\3,570

広島－和気 189.9km／\3,260

クエリー

\*\*\*広島(3143)から下り方向で200km到達地点

select l1.sales\_km-l2.sales\_km as sales\_km, l1.station\_id

from t\_lines l1 join t\_lines l2

where l1.line\_id=139 and l2.line\_id=139

and l1.sales\_km>l2.sales\_km

and l2.station\_id=3143

and (l1.sales\_km-l2.sales\_km)>2000 order by l1.sales\_km limit(1);

\*\*\*\*広島(3143)から上り方向で200km到達地点

select l2.sales\_km-l1.sales\_km as sales\_km, l1.station\_id

from t\_lines l1 join t\_lines l2

where l1.line\_id=139 and l2.line\_id=139

and l2.sales\_km>l1.sales\_km

and l2.station\_id=3143

and (l2.sales\_km-l1.sales\_km)>2000 order by l1.sales\_km desc limit(1);

# ■ 規則157条2項

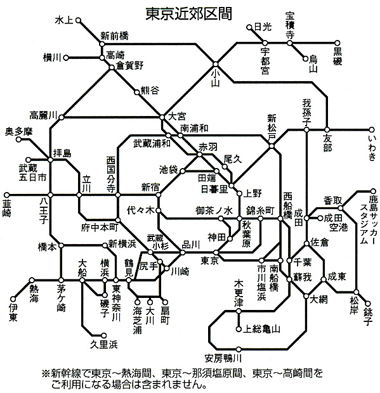
第157条2項で定める大都市近郊区間内発着の場合、最短経路で計算し、有効日数は1日となる（ただし、新幹線利用時はこのルールは適用されない）

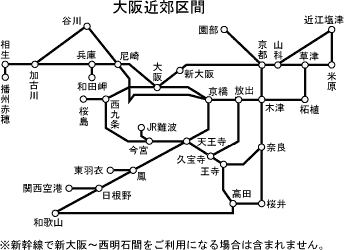
発駅または着駅が下記近郊区間外であっても、70条適用駅を通過した場合は、本規則が適用される。この場合、70条適用駅を2回通過した場合無効で、1回通過した場合は、70条適用駅の外端駅同士の最短距離を得て計算する。

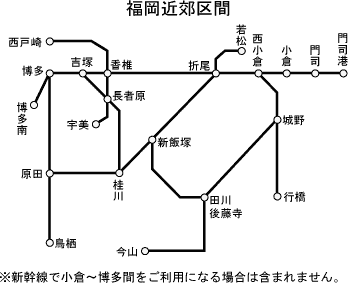
途中下車を望む場合、新幹線を経由に含め非適用にする。この場合の有効日数は乗車経路の営業キロとする。

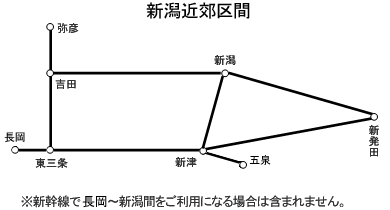
## データ定義

|  |  |  |
| --- | --- | --- |
| テーブルt\_station  city\_flg | | |
| BCSUBURB | 近郊区間   |  |  | | --- | --- | | 1 | 東京 | | 2 | 新潟 | | 3 | 大阪 | | 4 | 福岡 | | 4ビット |









## 実装詳細

経路がすべて同一近郊区間内である場合、適用される（新幹線など除外条件あり）。

（ 最短 ボタンが有効になる）

運賃計算時にチェックされる（一番最初におこなう）。

発駅～着駅までの最短距離をダイクストラ法で算出する。

伊東―水上

|  |  |  |
| --- | --- | --- |
| 東海道、相模、横浜、八高 | 260.5km( 269.7km） | \4620 |
| 東海道、相模、横浜、八高、高麗川、川越、大宮 | 300.5km(303.6km） | \5250 |
| 東海道、相模、横浜、中央東、武蔵野、埼京、大宮 | 286.7km | \4940 |
| 東海道、相模、横浜、中央東、山手、埼京、大宮 | 303.3km | \5250 |
| 東海道、東北 | 285.6km | \4940 |

※ DFS、SWA-MARSは4940、乗換案内は4620

ノード作成は分岐駅のみ

中間駅は全方向の分岐駅までのコストを得る。

得られた、全分岐駅からの最短経路を算出して決定

全ノード、未完、最小コスト無限に

開始ノードの最小コスト= 0

無限ループ

全ノードに対し、未完でかつ最小コストが無限でない最小コストが最も少ないノード番号を得る:doneNode

全て完了（ノード番号得ていない）場合、終了

得られたノードdoneNodeを完了に

ノードdoneNodeの接続された全ノード：to

ノードtoの最小コスト←doneNodeの最小コスト＋toのコスト

ノードtoのfrom ← doneNode

ノードtoの最小コストが無限or doneNodeの最小コスト+toのコストがノードtoの最小コスト未満？

**クエリーA**

→ 最初は開始ノードの点が得られる

jroute.pyで実装SQLの最適化、特別なデータ作成することなく上記フローを実行したところ、i5 2.3GHzで1秒強

新幹線対策未

同一計算キロの場合、どちらかひとつのルートが選択されるがほかの候補の選択は不明である。

同一運賃でも計算キロ（営業キロ）の少ない方が選択され、速度は全く考慮していないため、東京―大阪間は関西本線経由が選択されるし、函館―札幌も函館線経由が選択される。

Start

B

A

C

F

E

D

3

2

4

4

1

3

3

1

⓪

3A

2A

4A

5B

6E

経路探索は分岐駅のみでおこなわれる。

関数dijkstra()を終了するとStartからのすべての最短距離が全分岐駅にマークされる。

上記の図において、丸囲みのアルファベットはNameOfNode、数値はCostOfSection、CostOfRouteであり、CostOfRouteと一緒に記載されているアルファベットはFromNodeをしめします。

関数dijkstra(StartOfNode)実行後、最終ノード(EndOfNode)は任意のノードを選択できます。

上図でにおいて、F点を着駅とすると、F点をリストの先頭へ追加、次にF点に隣接している全ノードの最短コストノードはE点であるので、E点をリストへ追加、同様にE点は5Bであり、隣接ノードはコスト5でB点なのでB点をリストへ追加、さらに同様にA点を追加して終了（A点＝発駅＝開始点）。リストを逆順に並べ替えると最短経路となる。

分岐駅以外の駅の開始、終了は以下の方法となります。

* + 開始駅が分岐駅以外の時
    - 開始駅の隣接する両端の分岐駅A,Bを得る（一つしか得られなかった場合そのひとつのみ対象）、
    - A、Bの最小コストを開始駅からそれぞれAまたはBまでの計算キロとして初期化する（上図HCPチャートの「開始ノードの最小コスト= 0」と記述された部分）
    - A、BのFromNodeを開始駅に設定（上図HCPチャートの「ノードtoのfrom ← doneNode」の部分）
    - ※ done(完了)フラグはONにはしない
    - 以上の初期化後、関数dijkstra()を実行する（上図HCPチャート）
  + 終了駅が分岐駅以外の時
    - dijkstra()を実行
    - 終了駅の隣接する両端の分岐駅A、Bを得る（一つしか得られなかった場合、その駅を終了駅とする）
    - 終了駅からそれぞれAまたはBまでの計算キロをAc、Bcとする
    - (Aの最小コスト＋Ac)＜（Bの最小コスト＋Bc）なら、終了駅をA、ElseならBとなります（最小コストはdijkstra()実行後に得られている）

dijkstra(開始駅)実行後、終了駅からfromNodeを開始駅まで辿れば、最短ルートが得られる。

駅から路線を得る

select line\_id from t\_lines where station\_id=?

路線、駅から隣接する両端の分岐駅とそこまでの計算キロを得る（?1：路線／?2：駅）（クエリーA）

select station\_id, (select case when calc\_km>0 then calc\_km else sales\_km end from t\_lines where line\_id=?1 and station\_id=?2)-case when calc\_km>0 then calc\_km else sales\_km end as cost

from t\_lines

where line\_id=?1

and sales\_km=(select max(sales\_km)

from t\_lines l

join t\_station t

on t.rowid=l.station\_id

where line\_id=?1

and jctflg<>0

and sales\_km<(select sales\_km

from t\_station t, t\_lines l

where t.rowid=l.station\_id

and jctflg<>0

and line\_id=?1

and station\_id=?2))

union

select station\_id, case when calc\_km>0 then calc\_km else sales\_km end - (select case when calc\_km>0 then calc\_km else sales\_km end from t\_lines where line\_id=?1 and station\_id=?2) as cost

from t\_lines

where line\_id=?1

and sales\_km=(select min(sales\_km)

from t\_lines l

join t\_station t

on t.rowid=l.station\_id

where line\_id=?1

and jctflg<>0

and sales\_km>(select sales\_km

from t\_station t, t\_lines l

where t.rowid=l.station\_id

and jctflg<>0

and line\_id=?1

and station\_id=?2))

路線、駅から隣接する両端の分岐駅とそこまでの計算キロを得る（?1：路線／?2：駅）（クエリーA）

若干高速版

select id, (select case when calc\_km>0 then calc\_km else sales\_km end from t\_lines where line\_id=?1 and station\_id=?2)-case when calc\_km>0 then calc\_km else sales\_km end as cost

from t\_lines l join t\_jct j on j.station\_id=l.station\_id

where line\_id=?1

and sales\_km=(select max(sales\_km)

from t\_lines l

join t\_jct j

on j.station\_id=l.station\_id

where line\_id=?1

and sales\_km<(select sales\_km

from t\_lines

where line\_id=?1

and station\_id=?2))

union

select id, case when calc\_km>0 then calc\_km else sales\_km end - (select case when calc\_km>0 then calc\_km else sales\_km end from t\_lines where line\_id=?1 and station\_id=?2) as cost

from t\_lines l join t\_jct j on j.station\_id=l.station\_id

where line\_id=?1

and sales\_km=(select min(sales\_km)

from t\_lines l

join t\_jct j

on j.station\_id=l.station\_id

where line\_id=?1

and sales\_km>(select sales\_km

from t\_lines

where line\_id=?1

and station\_id=?2))

HCPチャートのクエリーAの実行がダイクストラの速度のボトルネックとなる。

分岐駅（t\_jctのレコード）の数は310件程度で大したことはないので最適化しなくても処理速度はかからないが、上記のクエリーAは970回実行される部分で処理速度の大半を占める。

この結果をDBに保持しておくことにより（t\_jctテーブルにnext\_id, cost列を追加）処理の速度の高速化が図れる。

python実装の以下の部分(疑似コード)の結果をテーブル作成

for jct\_id in (select id from t\_jct):

     jct\_id, cost, line\_id in node\_next(jct\_id)

クエリーAの部分は、以下クエリーで得る（pythonコードのnode\_itemに相当）

--　jct\_idと接続する隣の分岐駅リストを得る

select neer\_id, cost, line\_id from t\_node where jct\_id=:?

指定駅が分岐駅か

select jctflg from t\_station where rowid=?1

隣接する両端の分岐駅と計算キロを得る

?1 駅

select station\_id , abs((select case when calc\_km>0 then calc\_km else sales\_km end from t\_lines where line\_id=(select line\_id from t\_lines where station\_id=?1) and station\_id=?1)-case when calc\_km>0 then calc\_km else sales\_km end) as cost

from t\_lines

where line\_id=(select line\_id from t\_lines where station\_id=?1)

and sales\_km in ((select max(y.sales\_km)

from t\_lines x, t\_lines y, t\_station t

where x.line\_id=y.line\_id

-- and x.line\_id=(select line\_id from t\_lines where station\_id=?1)

and x.station\_id=?1

and t.rowid=y.station\_id

and t.jctflg<>0

and x.sales\_km>y.sales\_km) ,

(select min(y.sales\_km)

from t\_lines x, t\_lines y, t\_station t

where x.line\_id=y.line\_id

-- and x.line\_id=(select line\_id from t\_lines where station\_id=?1)

and x.station\_id=?1

and t.rowid=y.station\_id

and t.jctflg<>0

and x.sales\_km<y.sales\_km));

# その他規則

規則88条

大阪・新大阪－姫路以遠は、大阪から計算する。

規則89条

北新地から尼崎以遠へは大阪から計算する(料金のみ。有効期間は通常通り)

→ 個別判定し、自動チェックして計算

九州内山陽新幹線

* 臨時駅（臨）という接頭辞の駅は、その先までの駅の営業キロで計算するルールの適用を考慮すること

# ■ 会社線通過連絡運輸

会社線内駅で始発、終着することはない（できない）ので、分岐駅のみの運賃表を定義する。

テーブルt\_clinfarに、各路線につき、N=路線内の分岐駅数とすると、N×（N-1）／2とおりの運賃表レコードを定義する（以下例）。

|  |  |  |
| --- | --- | --- |
| 青森 | 野辺地 | 1010 |
| 青森 | 八戸 | 2220 |
| 青森 | 二戸 |  |
| 青森 | いわて沼宮内 |  |
| 青森 | 好摩 |  |
| 青森 | 盛岡 |  |
| 野辺地 | 八戸 |  |
| 野辺地 | 二戸 |  |
| 野辺地 | いわて沼宮内 |  |
| 野辺地 | 好摩 |  |
| 野辺地 | 盛岡 |  |
| 八戸 | 二戸 |  |
| 八戸 | いわて沼宮内 |  |
| 八戸 | 好摩 |  |
| 八戸 | 盛岡 |  |
| 二戸 | いわて沼宮内 |  |
| 二戸 | 好摩 |  |
| 二戸 | 盛岡 |  |
| いわて沼宮内 | 好摩 |  |
| いわて沼宮内 | 盛岡 |  |
| 好摩 | 盛岡 |  |

IGRいわて銀河、青い森鉄道は2線合わせて「IGRいわて銀河・青い森鉄道」と扱う

本当は、盛岡－二戸－目時：IGRいわて銀河、目時－八戸－青森：青い森鉄道

通過するのに、

|  |  |
| --- | --- |
|  | 新青森 |
| 東北線 | 青森 |
| 青い森鉄道 | 目時 |
| IGRいわて銀河 | 盛岡 |
| 東北線 | 北上 |

としなくてよく、

|  |  |
| --- | --- |
|  | 新青森 |
| 東北線 | 青森 |
| IGRいわて銀河・青い森鉄道 | 盛岡 |
| 東北線 | 北上 |

select dfare from t\_clinfar where cline\_id=?1 and ((station\_id1=?1 and station\_id2=?2) or (station\_id1=?2 and station\_id2=?1))

# ■ 運賃計算

## データ定義

|  |  |  |
| --- | --- | --- |
| テーブルt\_line  spe\_flg | | |
| BSRBORDER | 境界駅 | 1ビット |

2駅間の距離

旧

select sales\_km, calc\_km, spe\_route from t\_lines where line\_id=? and (station\_id=? or station\_id=?)

新

select l1.sales\_km - l2.sales\_km, l1.calc\_km - l2.calc\_km

from t\_lines l1

join t\_lines l2

where l1.line\_id=?1

and l2.line\_id=?1 -- 無くてもいい

and l1.sales\_km>l2.sales\_km

and ((l1.station\_id=?2 and l2.station\_id=?3)

or (l1.station\_id=?3 and l2.station\_id=?2))

※ 計算キロの場合1行目を以下に変更

select case when l1.calc\_km<>l2\_calc\_km then l1.sales\_km - l2.sales\_km else l1.calc\_km - l2.calc\_km end

運賃計算はDBではなく、計算により算出する。

計算により算出が難しい、上限が定義できる範囲はDBテーブルに定義する。

テーブルのフィールドは、下限[km]と料金[fare]の2フィールドで以下のようなクエリーで得る

select fare\_X from t\_fareX where km<=? order by km desc limit(1);

11, 16, 21, 30

11~15.9, 16~20.9, 21~20.9, 30~

* kmは0から定義すること
* kmの最後は最後に定義された下限値の上限値+1を定義し、fareは、0とする（これ以上は定義なしを意味する）

運賃計算は経路配列より以下の乗車キロ累積を計算する

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 路線 | 駅 | 営業キロ | 計算キロ | 会社種別 |
|  | あいの里公園 |  |  |  |
| 札沼線 | 桑園 |  |  |  |
| 函館線 | 白石 |  |  |  |
| 千歳線 | 沼ノ端 |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | 営業キロ | 計算キロ |
| 会社線を除くすべて |  |  |
| JR北海道部分 |  |  |
| JR四国部分 |  |  |
| JR九州部分 |  |  |

をそれぞれ算出

本州3社と三島会社の境界駅の定義

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **会社** | **路線** | **境界駅** | **路線** | **会社** |
| JR北海道 | 津軽線 | 中小国 | 津軽海峡線 | JR東日本  JR西日本 |
| JR四国 | 瀬戸大橋線 | 児島 | 瀬戸大橋線 |
| JR九州 | 山陽線 | 下関 | 山陽線 |
| 鹿児島・日豊線 | 小倉 | 山陽新幹線 |
| 鹿児島・九州新幹線 | 博多 | 山陽新幹線 |

上記で分岐駅以外の境界駅は、

* 児島（本四備讃線・茶屋町―宇多津間）
* 下関（山陽線・幡生―門司間）

四国

|  |  |  |  |
| --- | --- | --- | --- |
|  | 岡山 | JR西日本 |  |
| 宇野線 | 茶屋町 | JR西日本 |  |
| 本四備讃線 | 宇多津 | JR四国 | 児島 |
| 予讃線 | 高松 | JR四国 |  |

九州

|  |  |  |  |
| --- | --- | --- | --- |
|  | 博多 | JR九州 |  |
| 鹿児島線 | 門司 | JR九州 |  |
| 山陽線 | 櫛ヶ浜 | JR西日本 | 下関 |
| 岩徳線 | 岩国 | JR西日本 |  |

九州（新幹線‐博多）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 熊本 | JR九州 |  |
| 九州新幹線 | 博多 | JR九州 | 博多 |
| 山陽新幹線 | 新大阪 | JR西日本 |  |

九州（新幹線‐小倉）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 新大阪 | JR西日本 |  |
| 山陽新幹線 | 小倉 | JR九州 |  |
| 鹿児島線 | 西小倉 | JR九州 |  |
| 日豊線 | 大分 | JR九州 |  |

分岐駅≠境界駅の場合、三島―本州会社のまたがる路線の駅間にある境界駅を得る。

cityflg.BCOSMSP（大阪電車特定区間）、cityflg.BCTKMSP（東京電車特定区間）

### **運賃計算種別**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **記号** | **項目** | **時刻表 JTB/JR** | **適用距離** | **キロ種別** | **算出方法\*1** |
| a | 本州三社‐幹線 | A-1/A | 営業キロ＋運賃計算キロ | （I） | 処理内ハードコーディングfare\_a() |
| b | 東京電車区間 | D | 営業キロ | 処理内ハードコーディングfare\_b() |
| c | 大阪電車区間 | D | 営業キロ | 処理内ハードコーディングfare\_c() |
| d | 山手線内 | E | 営業キロ | 処理内ハードコーディングfare\_d() |
| e | 大阪環状線内 | E | 営業キロ | 処理内ハードコーディングfare\_e() |
| f | JR北‐幹線 | C-1/E | 運賃計算キロ | テーブル：t\_farem.f, g, h  上限超えは演算：fare\_f(), fare\_g(), fare\_h |
| g | JR四国 | C-1/G | 運賃計算キロ（営業キロ＋擬制キロ） |
| h | JR九州 | C-1/G |
| i | 本州三社‐地方交通線 | B-1/B | 営業キロ | (II) | テーブルt\_farel |
| j | JR北‐地方交通線 | B-1/F | 営業キロ |
| k | JR四国‐地方交通線のみの特例 | C-2/H | 擬制キロ  営業キロ | (III) | テーブルt\_farels |
| l | JR九州‐地方交通線のみの特例 | C-2/H | 擬制キロ  営業キロ |
| m | JR四国の幹線+地方交通線の特例  営業キロ10km運賃計算キロ11km以内 | C-3/I | 運賃計算キロ  営業キロ | (IV) | 処理内ハードコーディング  ~~→不要(k, l)対応で適用されるので~~ |
| n | JR九州の幹線+地方交通線の特例  営業キロ10km運賃計算キロ11km以内 | C-3/I | 運賃計算キロ  営業キロ |
| o | JR東＋JR北海道の加算額 | A-2/C | 運賃計算キロ | (V) | t\_fareopq |
| p | JR西＋JR四国の加算額 | A-2/C | 運賃計算キロ |
| q | JR西＋JR九州の加算額 | A-2/C | 運賃計算キロ |
| r | JR東＋JR北海道（江差線・海峡線のみ） | B-2/D | 営業キロ | (VI) | t\_farer |
| s | 南千歳-新千歳空港 | 加算運賃 | 140 | － | t\_farest |
| 日根野-関西空港 | 加算運賃 | 210 | － |
| 日根野-りんくうタウン | 加算運賃 | 150 | － |
| りんくうタウン-関西空港 | 加算運賃 | 160 | － |
| 児島-宇多津 | 加算運賃 | 100 | － |
| 田吉-宮崎空港 | 加算運賃 | 120 | － |
| t | その他都市部私鉄競合区間 | 特定運賃 |  | － |
| u | 会社線 | 「■ 会社線通過連絡運輸」参照 | | | t\_clinfar |

\*1 算出関数は、int fare\_a() 関数とする

### **運賃計算ロジック**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| if 本州3社あり？ | | | | | | | | |
|  | if 地方交通線のみ? | | | | | | | |
|  | (i)<s> | | | | 地方交通線のみ | | |
| else | | | | | 幹線のみ、幹線+地方交通線 | | |
|  | if 幹線のみ電車特定区間? | | | |  | | |
|  | return; | | | | | |
| else | | | |  | | |
|  | (a)<s><c> 幹線のみ、幹線+地方交通線  ※「幹線+地方交通線」で全体の営業キロが10km以下は(i)<s>で計算 | | | | | |
| if JR北海道あり? | | | | |  | | |
|  | if JR北海道側は地方交通線のみ? | | | | | | |
|  | JR北海道側(r)<s>加算 | | | | | |
| else | | | 幹線のみ、幹線+地方交通線 | | | |
|  | JR北海道側(o)<s><c>加算  ※「幹線+地方交通線」で全体の営業キロが10km以下は(r)<s>で計算 | | | | | |
| if JR九州あり? | | | | |  | | |
|  | JR九州側(q)<s><c> 加算 | | | | 幹線のみ、幹線+地方交通線 | | |
| if JR四国あり? | | | | |  | | |
|  | 四国側(p)<s><c> 加算 | | | | 幹線のみ、幹線+地方交通線 | | |
| else if JR北海道 | | | | | | | | |
|  | if 地方交通線のみ | | | JR北海道のみ(JR九州、四国ありはあり得ない) | | | | |
|  | (ｊ)<s> | |  | | | | |
| else | | | 幹線のみ、幹線+地方交通線 | | | | |
|  | (ｆ)<c> | |  | | | | |
| else if JR九州 | | | | | | | | |
|  | (h)<s><c> | | | | | | | JR九州のみ(JR北海道、JR四国ありはあり得ない) |
| if <s><c>が200㎞以下で地方交通線のみ? | | | | | | |
|  | (l)<s><c> | | | | | 地方交通線のみ |
| else | | | | | |  |
|  | (n)<s><c>  ※10km以下適用チェック\*1 | | | | | 幹線のみ、幹線+地方交通線 |
| else if 四国 | | | | | | | | |
|  | (g)<s><c> | | | | | | | JR四国(JR北海道、JR九州ありはあり得ない) |
| if <s><c>が200㎞以下で地方交通線のみ? | | | | | | |
|  | (k)<s><c> | | | | | 地方交通線のみ |
| else | | | | | |  |
|  | (m)<s><c>  ※10km以下適用チェック \*1 | | | | | 幹線のみ、幹線+地方交通線 |
| else | | | | | | | | |
|  | 空 あり得ない | | | | | | | |

()内・・・運賃計算表

<s> ・・・ 営業キロ　　　　　<c> ・・・ 計算キロ

\*1 地方交通線のみの場合は擬制キロと営業キロ、地方交通線＋幹線の場合は運賃計算キロと営業キロ

# ■ 有効日数計算

有効日数は以下の式で算出

#!python3.0.1

# -\*- coding: utf-8 -\*-

# 有効日数

# 営業キロ（×10して小数点第１位を整数）から有効日数を得る

def days\_ticket(km):

if k < 1001: # 100㎞以下は1日

return 1

d = (km + 1999) // 2000 + 1

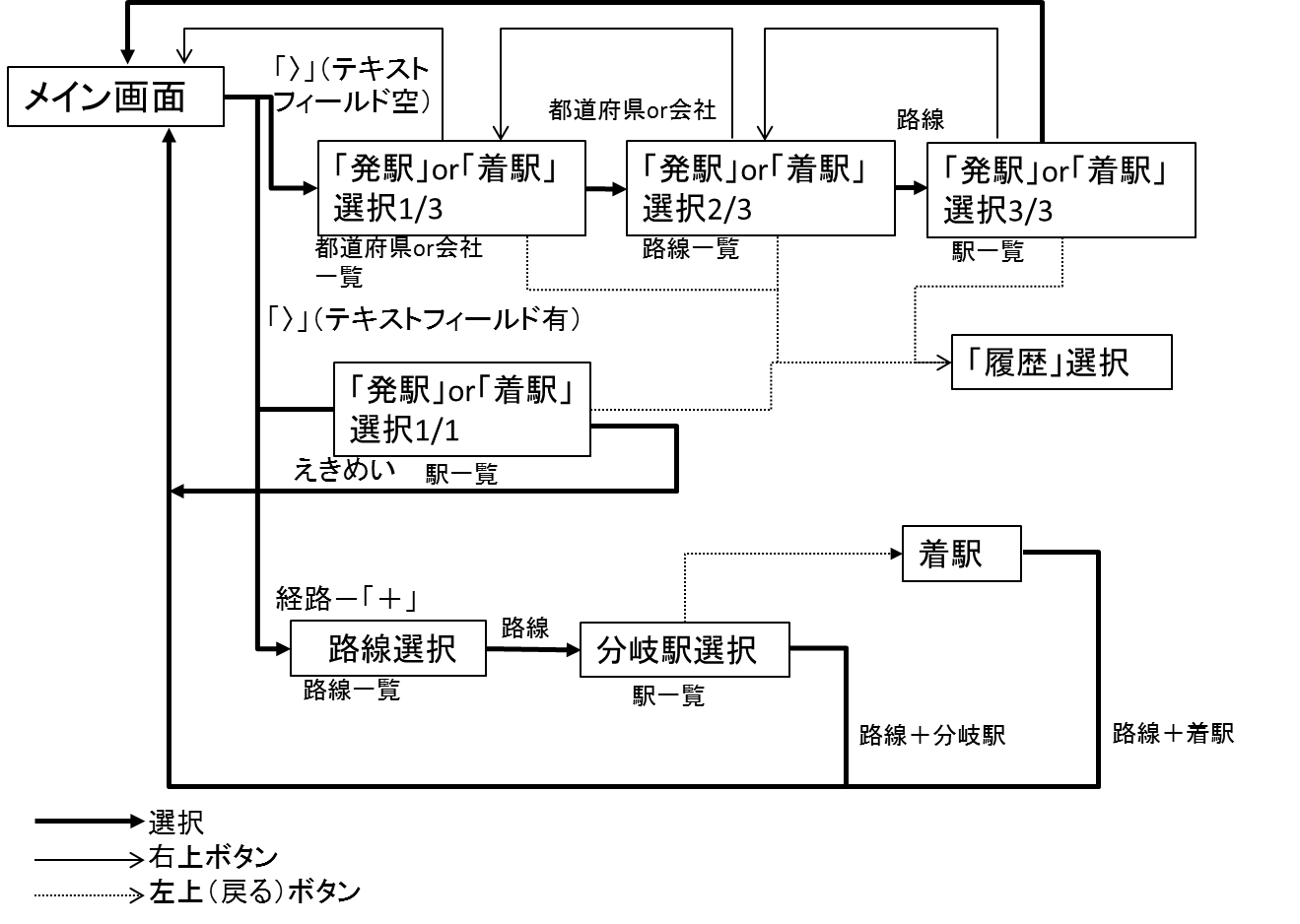
return d

但し、大都市近郊区間内で新幹線を利用しない最短経路運賃計算区間では1日とする。

# MVCインタフェース

## ファーストビュー

画面を最初に表示する際には、Routeオブジェクトの構築と初期化をおこなう。構築はコンストラクタによるアプリケーション起動時におこない、初期化も同時でも構わない。但し画面が作成されるたびには初期化メソッドとして、Route.clear()は呼び出す必要がある。



1発駅選択

2駅名（かな or カナ）から駅一覧

3都道府県から路線一覧

6駅から路線一覧

5路線から駅一覧

4都道府県+路線から駅一覧

直接入力

駅選択完了

都道府県選択解除

6駅から路線一覧

7路線選択→line

8路線から分岐駅一覧

9分岐駅・着駅選択

インデックス0に「着駅を選択する」を表示

「着駅を選択する」を選択

5路線から駅一覧

着駅選択→station2

インデックス0に「分岐駅を選択する」を表示」

分岐駅選択→station2

運賃表示

「分岐駅を選択する」を選択

route.add(line, station1, station2)

ルートチェック

↓

station1

station2→station1

## 1.発駅選択

route.setStartStation()

駅から路線一覧

「駅名から駅名リスト」、「都道府県から路線一覧」、「都道府県＋路線から駅一覧」、「路線から駅一覧」、「駅から路線一覧」使用

## 2.駅名から駅名リスト

概要

入力

出力

クエリー

説明（次に想定されるアクションなど）

## 3.都道府県から路線一覧

概要

入力

出力

クエリー

説明（次に想定されるアクションなど）

## 4.都道府県+路線から駅一覧

概要

入力

出力

クエリー

説明（次に想定されるアクションなど）

## 5.路線から駅一覧

概要

入力

出力

クエリー

説明（次に想定されるアクションなど）

Y字分岐のEx.1の例としてspe\_route.BSRJCTSPがONの場合、stationIdのBSRJCTSPをONに設定する。

## 6.駅から路線一覧

概要

入力

出力

クエリー

説明（次に想定されるアクションなど）

Y字分岐のEx.2の例のように「旭川」から「路線一覧」を表示する際、lineIdのbit30にspe\_route.BSRJCTSPがONの情報をコピーする。→route.add()を呼ぶときに、lineId.b30をstationId.b30へコピーし、lineId.b30=0にして渡す

## 7.路線選択

概要

route.setStartStation()

路線から分岐駅一覧

## 8.路線から分岐駅一覧

概要

リスト内のトップインデックスに「全駅（着駅）表示」項目を先頭に。選択すると「路線から駅一覧」を表示

入力

出力

クエリー

説明（次に想定されるアクションなど）

## 9.分岐駅／着駅選択

概要

route.add()

route.calcFare()

「6駅から路線一覧」表示

## 削除選択

Route.removeTail()をコール

そのあと、Route.iterator()をコールし、RouteIteratorオブジェクトを取得して表示を更新

## クリア選択

Route.clear()をコール

## 運賃計算

運賃計算は常時おこなう

＜チェック順＞

分岐特例補正<151>

重複チェック

山手線内発着<87>→自動にしない？（最後に適用可否調査後、選択ボタンで適用？）

都区市内発着<86>→自動にしない？（最後に適用可否調査後、選択ボタンで適用？）自動なら69の後？

特定市内発着料金逆転補正<114>(86の直後)

新幹線<->在来線乗換

東京環状線通過<70条>

特定選択経路<69条>

都市近郊区間最短経路

↓

運賃計算ロジック

特別料金

3島

地方交通線のみ

特定電車区間

特定私鉄競合区間

特定区間

# 未整理

2011-7-29

select sales\_km, calc\_km, cityflg, company\_id from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_company c on c.rowid=t.company\_id where line\_id=? and (station\_id=? or station\_id=?)";

===============================================================

----------------

query 一覧

No.summaryinout1/\*

1会社一覧を得るなし会社\*

2都道府県一覧を得るなし都道府県\*

3会社から路線会社路線\*

4都道府県から路線都道府県路線\*

5漢字から駅駅名-駅\*

6ひらがなから駅かな駅\*

7路線から駅路線駅\*

8路線から分岐駅路線駅\*

9都道府県+路線から駅都道府県,路線駅1

10会社+路線から駅会社,路線駅1

11駅から路線駅路線\*

12-------

13駅から駅の営業キロ、換算キロを得る　駅1, 駅2営業キロ,換算キロ1単一路線

14駅名+路線から駅コード駅,路線駅Id1

1select name from t\_compnay;

2select name from t\_prefect;

3select distinct(line ) from lines where company=? order by line; 'JR東海'

select n.name, t.company\_id from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id join t\_company c on c.rowid=t.company\_id where c.name='JR東海' group by l.line\_id order by n.name;

select n.name, line\_id from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id where company\_id=3 group by l.line\_id order by n.name;

4select distinct(line) from lines where prefect=?;'滋賀県'

select n.name, l.line\_id from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id join t\_prefect p on p.rowid=t.prefect\_id where p.name='佐賀県' group by l.line\_id;

select n.name, l.line\_id from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id where prefect\_id=41 group by l.line\_id order by n.name;

5select rowid, name from [t\_station] where name like ?:'北千%';

6select rowid, name from [t\_station] where kana like ?;'きた%';

7select station\_id, t.name from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id where n.name='小海線' order by sales\_km

8select station\_id, t.name from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id where t.jctflg<>0 and n.name='函館線' order by l.sales\_km

9select station\_id, t.name from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id join t\_prefect p on p.rowid=t.prefect\_id where n.name='小海線' and p.name='長野県' order by sales\_km

select station\_id, t.name from t\_lines l join t\_station t on t.rowid=l.station\_id where line\_id=90 and prefect\_id=5 order by sales\_km;

10select station\_id, t.name from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id join t\_company c on c.rowid=t.company\_id where n.name='東海道線' and c.name='JR東日本' order by sales\_km

select station\_id, t.name from t\_lines l join t\_station t on t.rowid=l.station\_id where line\_id=90 and company\_id=5 order by sales\_km;

11select line from lines where station=?;

select n.name, l.line\_id from t\_line n join t\_lines l on n.rowid=l.line\_id where station\_id=508;

12--------

13select sales\_km, calc\_km from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_line n on n.rowid=l.line\_id where (t.name='追分' or t.name='米沢' )and n.name='奥羽線' order by sales\_km desc;

14select l.station\_id from t\_lines l join t\_station t on l.station\_id=t.rowid join t\_line n on n.rowid=l.line\_id where t.name='大久保' and n.name='中央東線'

select distinct(line ) from lines where company=? order by line; 'JR東海'

select line FROM lines S1 WHERE EXISTS ( SELECT \* FROM lines S2 WHERE S1.key\_1 = S2.key\_1 AND S1.key\_2 = S2.key\_2 AND S1.rowid < S2.rowid)

select line FROM lines S1 WHERE EXISTS ( SELECT \* FROM lines S2 WHERE S1.line = S2.line AND S1.rowid < S2.rowid) where compay='JR西日本'

立川－高尾間の分岐駅(結果は、立川と八王子)一覧

select \* from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name='中央東線' and jctflg!=0 and

(sales\_km<=(select sales\_km from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name='中央東線' and t.name='高尾') and

sales\_km>=(select sales\_km from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name='中央東線' and t.name='立川')) or

(sales\_km>=(select sales\_km from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name='中央東線' and t.name='高尾') and

sales\_km<=(select sales\_km from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name='中央東線' and t.name='立川'))

↓コードのみ(中央東線line\_id=23, 立川=532、高尾=537

select id from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_jct j on j.station\_id=l.station\_id where line\_id=23 and jctflg!=0 and

((sales\_km<=(select sales\_km from t\_lines where line\_id=23 and station\_id=537) and

sales\_km>=(select sales\_km from t\_lines where line\_id=23 and station\_id=532)) or

(sales\_km>=(select sales\_km from t\_lines where line\_id=23 and station\_id=537) and

sales\_km<=(select sales\_km from t\_lines where line\_id=23 and station\_id=532)))

↓もっと実用的に(bind(1, line), bind(2, eki1), bind(3, eki2))

"select t.name from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name=?1 and jctflg!=0 and "

"((sales\_km<=(select sales\_km from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name=?1 and t.name=?2) and "

"sales\_km>=(select sales\_km from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name=?1 and t.name=?3)) or "

"(sales\_km>=(select sales\_km from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name=?1 and t.name=?2) and "

"sales\_km<=(select sales\_km from t\_lines l join t\_line n on n.rowid=l.line\_id join t\_station t on t.rowid=l.station\_id where n.name=?1 and t.name=?3)))"

select id,t.name from t\_lines l join t\_station t on t.rowid=l.station\_id join t\_jct j on j.station\_id=l.station\_id where line\_id=93 and jctflg!=0 and

((sales\_km<=(select sales\_km from t\_lines where line\_id=93 and station\_id=1996) and

sales\_km>=(select sales\_km from t\_lines where line\_id=93 and station\_id=2050)) or

(sales\_km>=(select sales\_km from t\_lines where line\_id=93 and station\_id=1996) and

sales\_km<=(select sales\_km from t\_lines where line\_id=93 and station\_id=2050)))

const char\* cmd\_sql\_linedup =

"select station\_id from t\_lines where line\_id=?1 and ("

"station\_id=?2 OR "

"station\_id=?3 OR "

"station\_id=?4 OR "

"station\_id=?5) order by sales\_km";

----

DBをプログラムからどのようにして読み出すか

　　Browfish暗号する

　　リソースファイルの保持

　　その形式は？

sqlite3

.backup file

db file create

sqlite3\_backup\_init() api used.

sqlite3\_backup\_xxx() api used.

.dump

sql statementエコーバック

.restore file

.backup file 's restore.と同様以下APIで実現

sqlite3\_backup\_init() api used.

sqlite3\_backup\_xxx() api used.

結論

　1- RCには、Dumpして生成された全SQL文をBrowfish暗号格納

　2- RCリード

　3- RC復号

　4- :memory: DBへCreate

　5- DBへRC復号SQL文を実行

　1．Dumpツールは？RCへの登録 ちょっとめんどいけど頻度は少ないはずだし手間もどうってことないはず。

　1. 3．鍵は？ CBC 初期ベクトル const変数ハードコード UUIDGEN、

　　UUIDGENの下4ビットのスクランブルの暗号のスクランブル。

　5．APIは？sql3\_exec()で一括

　Tempファイルを作成が簡単なのだが、拾われる恐れがある。

　-> Tempファイル作成にする。

　　作成して :memory: open -> sqlite3\_backup\_init()でrestoreして tempを削除

　　~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <--- 決定

　理想

　memoryをファイルとして扱えないか？

　メモリイメージをSQLITE側でアタッチできないか？

　uuidgenの256×2バイト

# Appendix.

# DB定義

### **テーブル**

### **t\_company 会社線テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_company**  **（会社線テーブル）** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | name | char(11) | pk | 会社名 |

### **t\_prefect 都道府県テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_prefect**  **（都道府県テーブル）** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | name | char(12) | pk | 都道府県名 |

### **t\_line 路線名テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_line**  **（路線名テーブル）** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | name | text |  | 路線名（駅名と重ならない名称は末尾の「線」を省略する） |

* 新幹線はテーブルの最終レコードに集められ、新幹線か在来線か否かについては最初の新幹線のidを固定なハードコード定義（#defineまたはfinal intなど)で指定し、その値以上は新幹線、未満は在来線と判断されるようにする。
* 上越新幹線、北陸長野新幹線は東京から定義され、東京－大宮は、東北新幹線と重複し、大宮－高崎間は、上越新幹線、北陸長野新幹線は重複する

### **t\_station 駅テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_station**  **（駅テーブル）** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | name | text |  | 駅名 |
|  | kana | text |  | 駅名読み（かな） |
|  | company\_id | int | t\_company(rowid) | 所属会社 |
| \* | prefect\_id | int | t\_prefect(rowid) | 都道府県名 |
|  | samename | text |  | 同名駅区別文字（同名駅でなければ空文字） |
|  | sflg | int |  | フラグ |

### **t\_lines 路線テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_lines**  **（路線テーブル）** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | line\_id | int | t\_line(rowid) | 路線   * 新幹線は在来線のあとに定義 * 通貨連絡運輸の会社線は最後半に定義 |
| \* | station\_id | int | t\_station(rowid) | 駅 |
|  | sales\_km | int |  | 下り起点駅からの営業キロ  (特例ルートによっては異なる情報が格納) |
|  | calc\_km | int |  | 下り起点駅からの計算キロ  (特例ルートによっては異なる情報が格納) |
|  | lflg | int |  | 特例ルートフラグ |

~~sales\_km=-1:新幹線で新幹線の駅の無い在来線分岐駅(平行在来線の新幹線にない分岐駅。名古屋、沼津とか新富士は含まず、金山、草津とか山科は含む。また国府津、富士、沼津などは異なる線なので含まない。※ 分岐駅の列挙「乗車経路マーク」時に使用~~

calc\_km=-1: 会社線通過連絡運輸の路線の場合

lflg.BSRJCTSP=1:分岐特例の定義で、sales\_km=分岐駅特例の実(戸籍上の)分岐駅(c) / calcl\_km=実路線(b)を表す。

### **t\_jct 分岐テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_jct**  **（分岐テーブル）** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | id | int | autoincrement |  |
|  | station\_id | int | t\_station(rowid) |  |

### **t\_node ノード定義テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_node**  **（分岐テーブル）**For dijkstra() | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | jct\_id | int | t\_jct.id |  |
| \* | neer\_id | int | t\_jct.id | jct\_idの隣の分岐駅 |
|  | cost | int |  | jct\_id ~ neer\_idの計算キロ |
| \* | line\_id | int | t\_line.rowid | 路線名 |

* jct\_idとneer\_idは入れ替わった場合も含めた組み合わせでユニーク

### **t\_clinfar 会社線駅間料金定義テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_clinfar**  **（会社線駅間料金定義テーブル）** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | station\_id1 | int |  | 駅1 |
| \* | station\_id2 | int |  | 駅2 |
|  | fare | int |  | 駅1～駅2の大人料金 |
|  | ~~line\_id~~ | ~~int~~ | ~~t\_line(rowid~~ |  |

### **t\_rule69 旅客営業規則69条適用テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_rule69**  **（規則69条適用テーブル）** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | id | int | 1~8 | 経路ID |
|  | station\_id1 | int | t\_station(rowid) | 発： |
|  | station\_id2 | int | t\_station(rowid) | 至： |
| \* | line\_id | int | t\_line(rowid) | 置き換え後の路線 |
|  | ord | tinyint |  | 表示順(下り) |

### **t\_rule86 旅客営業規則86条、87条定義テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | **t\_rule86**  **旅客営業規則86条、87条定義テーブル** | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | line\_id1 | int | t\_line(rowid) | 脱出（進入）路線 |
|  | station\_id | int | t\_station(rowid) | 乗換駅 |
|  | line\_id2 | int | t\_line(rowid) | 中心駅発路線 |
| \* | city\_id | int |  | 都区市内番号(1～11)  複数レコードの場合、2レコード目以降、+0x100, +0x200,・・・ |

* line\_id2の路線はcity\_idでしめす都区市内中心駅を有する路線である必要があります
* station\_idは、line\_id1にもline\_id2にも接続する駅である必要があります
* 脱出（進入）路線が、都区内中心駅から2路線を越える場合、同一のline\_id1の2レコード定義となる（おおさか東線）。この場合、city\_idのbit15-8までにレコード番号(2つめ=1即ち+0x10した値)が格納される。

### **t\_farem 3島会社幹線キロ程定義テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | t\_farem  キロ程定義テーブル（幹線、基準） | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | km | int |  | 運賃計算キロの下限 |
|  | f | int |  | JR北海道のキロ別料金(=0で上限越えをしめす) |
|  | g | int |  | JR四国のキロ別料金(=0で上限越えをしめす) |
|  | h | int |  | JR九州のキロ別料金(=0で上限越えをしめす) |

kmは0から定義すること

kmの最後は最後に定義された下限値の上限値+1を定義し、fareは、0とする（これ以上は定義なしを意味する）

### **t\_farel 地方交通線キロ程定義テーブル**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | t\_farel  キロ程定義テーブル（地方交通線のみ利用） | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | km | int |  | 運賃計算キロの下限 |
|  | i | int |  | 本州3社の地方交通線キロ別料金 |
|  | j | int |  | JR北海道の地方交通線キロ別料金 |

### **t\_farels 3島会社地方交通線キロ程定義テーブル特例**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | t\_farels  キロ程定義テーブル | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | dkm | int |  | 擬制キロ |
|  | skm | int |  | 営業キロ |
|  | k | int |  | JR四国適用料金 |
|  | l | int |  | JR九州適用料金 |

### **t\_fareopq 本州3社+3島会社加算**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | t\_fareopq  本州3社+3島通過加算テーブル | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | km | int |  | 運賃計算キロ |
|  | o | int |  | JR北海道加算額 |
|  | p | int |  | JR四国加算額 |
|  | q | int |  | JR九州加算額 |

### **t\_farer 本州3社+JR北海道地方交通線会社加算**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | t\_farer  JR東日本+JR北海道地方交通線通過加算テーブル | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | km | int |  | 営業キロ |
|  | r | int |  | JR北海道加算額 |

### **t\_farest 特定区間加算**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | t\_fares  特定区間通過加算テーブル | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | station1 | int |  | 発着駅1のstation\_id |
| \* | station2 | int |  | 発着駅2のstation\_id |
|  | fare | int |  | 料金 |
|  | kind | int |  | =0: 特定区間通過加算(s)  ≠0: 特定区間料金(t) |

### **t\_hzline 新幹線の並行在来線**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル名**  **（説明）** | | t\_hzline  新幹線の並行在来線 | | |
| **Key** | **列** | **型** | **制約** | **説明** |
| \* | line\_id | int | t\_line(rowid) | 並行在来線ID |

旧定義

spe\_route　の整理

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル** | | | **列** | |
| t\_lines | | | spe\_route | |
| **ビット** | **ラベル** | | **説明** | |
| 31 | BSRJCTSP | | 分岐駅特例, or <rule1>適用したらON,しない場合OFFにする | |
| 30 |  | | 0 (RouteItem挿入時、発駅側が分岐特例の場合にON(<rule3>)にする | |
| 29 | BSR69TERM | | 営業規則69条に該当する路線・分岐駅(端駅1、端駅2)に定義（中間駅には該当しない）。このビットがONの場合、bit3-0は営業規則69条の項番も設定される  ※ 「営業規則69条最短経路で計算」ルート検出に使用 | |
| 28 | BSR69CONT | | 営業規則69条に該当する路線・分岐駅(端駅1、端駅2)に定義（中間駅には該当しない）で、ONの場合は路線にまたがっており、続きがあることをしめします（東海道線、東海道新幹線、北陸線の米原に該当）  このビットがONの場合、bit3-0は営業規則69条の項番も設定される  ※ 「営業規則69条最短経路で計算」ルート検出に使用 | |
| 27 |  | | 新幹線の在来線分岐駅(平行在来線の新幹線にない分岐駅。名古屋、沼津とか新富士は含まず、草津とか山科は含む)  ※ 分岐駅の列挙「乗車経路マーク」時に使用 | |
| 26 |  | | 新幹線の平行在来線で乗換駅であるとON（在来線側のみで新幹線側は無効）  以下の25-24ビットとセットで使用される  ※ 新幹線の並行在来線取得に使用 | |
| 25-24 |  | | 平行在来線⇔新幹線乗り換え時の侵入許可フラグ（並行在来線側のみで新幹線側は無効）  00=通常（豊橋、浜松、新横浜、東京、新富士など）  01=新幹線降車後、新幹線下り線→平行在来線上り有効または、在来線から新幹線乗車時、在来線下り線→新幹線上り有効（静岡など）  10=新幹線降車後、新幹線上り線→平行在来線下り有効または、在来線から新幹線乗車時、在来線上り線→新幹線下り有効（名古屋、三島など）  11=上下線、平行在来線上下有効（広島など）  ※ 「新幹線と在来線乗り換え」で使用 | |
|  |  | | ~~ONで新幹線駅（平行在来線駅併設）新大阪とか（古川新花巻は、OFFでjctflgがON)~~ | |
|  |  | | ~~ONで新幹線の在来線分岐駅で分岐駅特例乗車可能区間の本来の分岐駅（例：金山）これがONの場合、ビット27もONである~~  ~~ONで新幹線駅(新幹線のみ。新幹線の沼津などはOFF)~~  未割当 | |
| 23 | BSR70 | | 規則70条定義レコード ONのとき、line\_id列は70条用に払い出し  station\_idは、b0～b15の路線の計算駅  b0～15は進入 or 脱出路線 | |
| 22 | BSRVIRTUAL | | 仮想駅（営業キロ計算用で、リスト表示時、分岐駅マーク時には適用しない駅） | |
| ~~21-20~~ | ~~BSRC114~~ | | ~~city\_flg.BCC114=1のときに有効~~  ~~00： 本路線では無効~~  ~~01: 下り線で有効~~  ~~10: 上り線で有効~~  ~~11: 上下線で有効（否定義）~~ | |
| ~~19-11~~ | ~~BSRD114~~ | | ~~BSRC114≠00のときのみ有効~~  ~~中心駅からの運賃計算キロをしめす。87条の場合、100km(1000)を引いた値、86条の場合、200km(2000)を引いた値(0～511)~~   * ~~100/200kmに達する駅が151.1/251.1kmを越える場合151.1/251.1kmに切り捨てて定義する（それでも可のはず）~~ | |
|  |  | | t\_station.jctflgの内容をここに含む | |
|  | BSRBORDER | | 境界駅 | |
|  |  | |  | |
| 新幹線の場合 | | | | |
| 8‐0 |  | | 0～511 並行在来線路線id | |
| 在来線の場合 | | | | |
|  |  | |  | |
|  |  | |  | |
| BSR70=1の場合 | | | | |
|  | |  | |  |
| 14-0 | | BSR70MASK | | 大都市環状線(70条適用駅)エリアへの進入 or 脱出路線 |
| 在来線の場合（bit29=1） | | | | |
|  |  | |  | |
| 3-0 | BSR69NOMASK | | 0～8 営業規則69条の項番に相当（経路ID）t\_rule69 | |

#define FLGTEST(flg, MASK) (flg & MASK)

#define FLGVALU(flg, MASK, POS) ((flg & MASK) >> POS)

// 一般路線

#define BSRNMLMASK (~(BSR70 | BSRVIRTUAL |

city\_flgの整理

|  |  |  |
| --- | --- | --- |
| **テーブル** | | **列** |
| t\_station | | cityflg |
| **ビット** | **ラベル** | **説明** |
|  |  |  |
| 31-21 |  | 空き |
|  |  | t\_station.jctflgを廃止してこのビットに割り当てる  さらにt\_lines.spe\_routeにも含ませる  0＝分岐駅なし、1＝分岐駅あり |
|  | BCOSMSP | 大阪電車特定区間 |
|  | BCTKMSP | 東京電車特定区間 |
| 20-17 | BCSUBURB | 近郊区間   |  |  | | --- | --- | | 1 | 東京 | | 2 | 新潟 | | 3 | 大阪 | | 4 | 福岡 | |
| 16 | BCRULE70 | 旅客営業規則70条適用駅 |
| 15 | BCYAMATE | 山手線内駅 |
|  |  | ~~都区市内代表駅からの営業キロ×10km(ビット4=1のときのみ有効、ビット4=0のときは無効)~~  ~~都区市内代表駅一覧は「~~**~~エラー! 参照元が見つかりません。~~**~~」参照~~ |
|  |  | ~~1のとき、b14-5の10ビットが都区市内センターからの営業キロ(範囲の再外側駅のみ)~~ |
|  | ~~BCC114~~ | ~~都区市内中心駅から100km、200kmに達する最初の駅（114条チェック用）(BCCITYNO≠0かつ、spe\_route.BSRC114≠0の路線でのみ有効)~~ |
| 4 | BCCITYCT | 都区市内中心駅の場合1（BCCITYNO≠0であること） |
| 3-0 | BCCITYNO | 都区市内番号（1～12）   |  |  |  | | --- | --- | --- | | 1 | 東京都区内 | 0x01 | | 2 | 横浜市内 | 0x02 | | 3 | 名古屋市内 | 0x03 | | 4 | 京都市内 | 0x04 | | 5 | 大阪市内 | 0x05 | | 6 | 神戸市内 | 0x06 | | 7 | 広島市内 | 0x07 | | 8 | 北九州市内 | 0x08 | | 9 | 福岡市内 | 0x09 | | 10 | 仙台市内 | 0x0a | | 11 | 札幌市内 | 0x0b | |

### **spe\_route(lflg)　の整理**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **テーブル** | | | | **列** |
| t\_lines | | | | spe\_route |
| **ビット** | | **ラベル** | | **説明** |
| 31 | | BSRJCTSP | | 分岐駅特例, or <rule1>適用したらON,しない場合OFFにする |
| 30 | | BSRJCTFLG | | 0 (RouteItem挿入時、発駅側が分岐特例の場合にON(<rule3>)にする  （このビットが1のときはDBエラーとする） |
| 29 | | BSR69TERM | | 営業規則69条に該当する路線・分岐駅(端駅1、端駅2)に定義（中間駅には該当しない）。このビットがONの場合、bit BSR69NOMASKは営業規則69条の項番も設定される  ※ 「営業規則69条最短経路で計算」ルート検出に使用 |
| 28 | | BSR69CONT | | 営業規則69条に該当する路線・分岐駅(端駅1、端駅2)に定義（中間駅には該当しない）で、ONの場合は路線にまたがっており、続きがあることをしめします（東海道線、東海道新幹線、北陸線の米原に該当）  このビットがONの場合、BSR69TERM は必ずONで、bit BSR69NOMASKは営業規則69条の項番も設定される  ※ 「営業規則69条最短経路で計算」ルート検出に使用 |
| 27 | | BSRSHINKTRS | | 新幹線の平行在来線で乗換駅であるとON（在来線側のみで新幹線側は無効）  以下の2ビットとセットで使用される  ※ 新幹線の並行在来線取得に使用   * 使用しなくても良いらしい |
| 26-25 | | BSRSHINKTRSALW | | 平行在来線⇔新幹線乗り換え時の侵入許可フラグ（並行在来線側のみで新幹線側は無効）  00=通常（豊橋、浜松、新横浜、東京、新富士など）  01=新幹線降車後、新幹線下り線→平行在来線上り有効または、在来線から新幹線乗車時、在来線下り線→新幹線上り有効（静岡など）  10=新幹線降車後、新幹線上り線→平行在来線下り有効または、在来線から新幹線乗車時、在来線上り線→新幹線下り有効（名古屋、三島など）  11=上下線、平行在来線上下有効（広島など）  ※ 「新幹線と在来線乗り換え」で使用 |
| 24 | | BSR70 | | 規則70条定義レコード ONのとき、line\_id列は70条用に払い出し  station\_idは、b0～b15の路線の計算駅  b0～15は進入 or 脱出路線 |
| 23 | | BSRCOMPANY | | 会社線のとき1 |
| 22 | | BSRVIRJCT | | 新幹線で新幹線の駅の無い在来線分岐駅(平行在来線の新幹線にない分岐駅。名古屋、沼津とか新富士は含まず、金山、草津とか山科は含む。また国府津、富士、沼津などは異なる線なので含まない。※ 分岐駅の列挙「乗車経路マーク」時に使用 |
| 21 | | BSRBORDER | | 本州3社と3島会社の運賃計算用の境界駅 |
| 20-17 | | BSR69NOMASK | | 0～15 ：営業規則69条の項番に相当（経路ID）t\_rule69  ※「在来線」で（BSR69TERM=1）の場合に適用 |
| 16-13 | | BSRHZLIN | | 0～15：新幹線の並行在来線IDテーブル（t\_hzlineのrowid）  （新幹線のみ） |
| BSR70=1の場合 | | | | |
| 15-0 | | BSR70MASK | | 大都市環状線(70条適用駅)エリアへの進入 or 脱出路線id |
| 上記以外 | | | | |
| 12-0 |  | | sflgの内容 | |

#define FLGTEST(flg, MASK) (flg & MASK)

#define FLGVALU(flg, MASK, POS) ((flg & MASK) >> POS)

// 一般路線

#define BSRNMLMASK (~(BSR70| ・・・)

### **city\_flg(sflg)の整理**

|  |  |  |
| --- | --- | --- |
| **テーブル** | | **列** |
| t\_station | | cityflg |
| **ビット** | **ラベル** | **説明** |
| -13 |  | Lflgの内容 |
| 12 | BCJCTFLG | 0＝分岐駅なし、1＝分岐駅あり |
| 11 | BCOSMSP | 大阪電車特定区間 |
| 10 | BCTKMSP | 東京電車特定区間 |
| 7-9 | BCSUBURB | 近郊区間   |  |  | | --- | --- | | 1 | 東京 | | 2 | 新潟 | | 3 | 大阪 | | 4 | 福岡 | |
| 6 | BCRULE70 | 旅客営業規則70条適用駅 |
| 5 | BCYAMATE | 山手線内駅/大阪環状線内（大阪か山手かはBCOSMSP/BCTKMSP参照） |
| 4 | BCCITYCT | 都区市内中心駅の場合1（BCCITYNO≠0であること） |
| 0-3 | BCCITYNO | 都区市内番号（1～12）   |  |  |  | | --- | --- | --- | | 1 | 東京都区内 | 0x01 | | 2 | 横浜市内 | 0x02 | | 3 | 名古屋市内 | 0x03 | | 4 | 京都市内 | 0x04 | | 5 | 大阪市内 | 0x05 | | 6 | 神戸市内 | 0x06 | | 7 | 広島市内 | 0x07 | | 8 | 北九州市内 | 0x08 | | 9 | 福岡市内 | 0x09 | | 10 | 仙台市内 | 0x0a | | 11 | 札幌市内 | 0x0b | |

CITYFLGはいれられるものはspe\_routeに含めることにより、t\_lines参照時、t\_stationを結合する手間を省く

（参考）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | |  | | | | | | | |  | | | | | | | |  | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |