Geocoding in Oracle using HERE Map Content in Oracle Delivery Format

A HERE White Paper

As more enterprises begin using maps for advanced analytics there Is a requirement to be able to assign locations (longitude/latitude) to addresses for display on maps and for analytic purposes such demographic analysis and understanding more about how locality/geography effect business . The address can be those of customers, warehouses, store locations, employees, and more. This paper describes how to assign geographic coordinates to addresses using the Oracle Spatial and Graph Geocoder with HERE Map Content.

By Dan Abugov
Business Development Manager and Consultant
HERE

Table of Contents

1 Introduction

Oracle continues to expand the functionality used for making maps and including location analytics in applications. The use of standard Oracle database components like Oracle Locator and Oracle Spatial and Graph to store and analyze geographic data continues to expand rapidly. Oracle Fusion Middleware MapViewer, used for map display and mashups, is included with every Oracle Application Server/Fusion Middleware License. These technologies enable the use of next generation mapping and location analysis capabilities in every Oracle application.

The geographic capabilities with Oracle match the needs of more and more businesses that require the capability to visualize the location of customers, shops, warehouses, employees, and more on a map, and, more importantly, to visualize the key performance indicators relating to these assets in a clear and concise way using maps. This is driving applications developers, who understand the need to easily add maps and into their applications. Oracle packaged applications such as OBIEE, Oracle Field Service, Oracle Utilties, Oracle Real-time Scheduler, Oracle Transportation Management, and hundreds of other applications including custom applications not created by Oracle have all been able to take advantage of the mapping and analysis capabilities of the Oracle platform. All of these applications are able to take advantage of HERE Map Content.

Oracle and Nokia have been collaborating to provide turn-key HERE Map Content and other content specifically created for the Oracle platform for over a decade. This mature content conforms to the Oracle Locator and Oracle Spatial and Graph standards for Mapping and Location Analysis, Geocoding, and Routing. HERE Map Contents include pre-built tables and indexes and installs in minutes using Oracle Transportable Tablespaces. HERE Map Content is utilized by all Oracle Applications that require maps and other location content, and are also used by many custom applications built for the Oracle platform.

This white paper focuses on using the HERE Map Content for geocoding in the Oracle Database.

2 Overview of Oracle's Location Technologies

Oracle Databases (Express, Personal, Standard and Enterprise Editions) all include built-in storage, index, and query capabilities for *location data*. We can define location data as coordinates that represent the location of a feature on the surface of the Earth. Using stored location information from an Oracle Database we have the ability to show features on a map as well as analyze features based on interactions and proximity to other features. This feature of the Oracle Database is called *Oracle Locator*.

Every Oracle Fusion Middleware license includes the ability to render maps for display in a wide variety of web-based applications. The map rendering component of the application server is called *Oracle Fusion Middleware MapViewer*.

Putting mapping aside momentarily, the built-in location capabilities in all Oracle Databases include the ability to analyze information geographically, allowing us to answer questions such as:

How many other hotels are within two miles of a proposed hotel location?
How many of my customers have average household income of greater than $67,000?
What is the closest ATM to my current location?

In addition to mapping and the abovementioned analysis capabilities, *Oracle Spatial and Graph* is an option with Oracle Enterprise Edition. Oracle Spatial and Graph includes advanced capabilities beyond the basic capabilities shown above. Oracle Spatial and Graph includes functionality such as Geocoding, Routing, Image Storage and retrieval, Network Analysis, Topology, Web Services, and a lot more.

If this paper is the first you have heard of Oracle's Location technologies, please see the Introduction to Mapping and Location Analysis in Oracle white paper:

http://download.oracle.com/otndocs/products/mapviewer/pdf/IntroductionToMappinginOracleUsingHERE_Maps.pdf

If after reading this white paper you wish to learn more about the Oracle Geocoder, go to:

http://download.oracle.com/otndocs/products/spatial/pdf/12c/orasptialandgraph_12c_wp_geocoder.pdf

3 Geocoding in Oracle using HERE Map Content: Hands-on

The following describes the setup steps required to follow this white paper as a "hands on" exercise.

Setup – HERE Map Content

This presentation shows the use of HERE Map Content. The data used in this presentation is available from the Oracle Technology Network . To download the data:

Go to:
http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/index.html

Click on the Downloads tab near the top of the page
Then click on Spatial Features Partners' Data Downloads

Click on "HERE Map Content Sample in ODF for San Francisco"
Click the Download link

After downloading, follow the included instructions for loading and setting up the data.

The sample data is the same data that is used by many Oracle Applications. All of the Oracle applications that require map content use HERE Map Content. A list of application that require HERE Map Content in Oracle Delivery Format and/or that can integrate HERE Map Content includes:

Oracle eLocation Services (http://maps.oracle.com/elocation)
Oracle E-Business Suite Field Service Advanced Scheduler
Oracle Utilities Mobile Workforce Management
Oracle Real-time Scheduler
Oracle Business Intelligence Enterprise Edition (OBIEE)
Oracle Transportation Management

Also note that many other applications easily integrate with the HERE Map Content and other content using the Oracle Database and Application Server.

4 Geocoding

Geocoding is the process of taking a text version of an address and returning the location of that address. Typically geocoding is done for one or a combination of three reasons:

- The geocoded locations will be displayed as points on a map
- The geocoded locations will be used for location analysis as shown previously (within distance, relationship, or nearest neighbor)
- The geocoded locations will include other information that is used as input to a router for generating driving (or other) directions

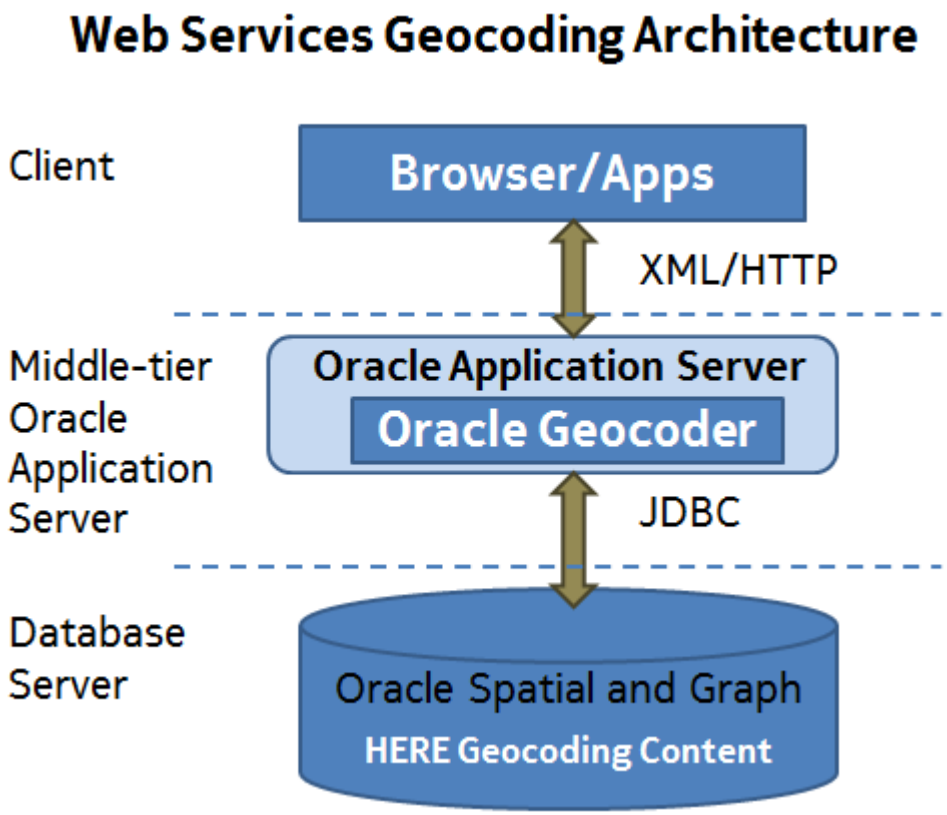
Figure 1 Geocoded addresses displayed on a map

5 Geocoding in Oracle

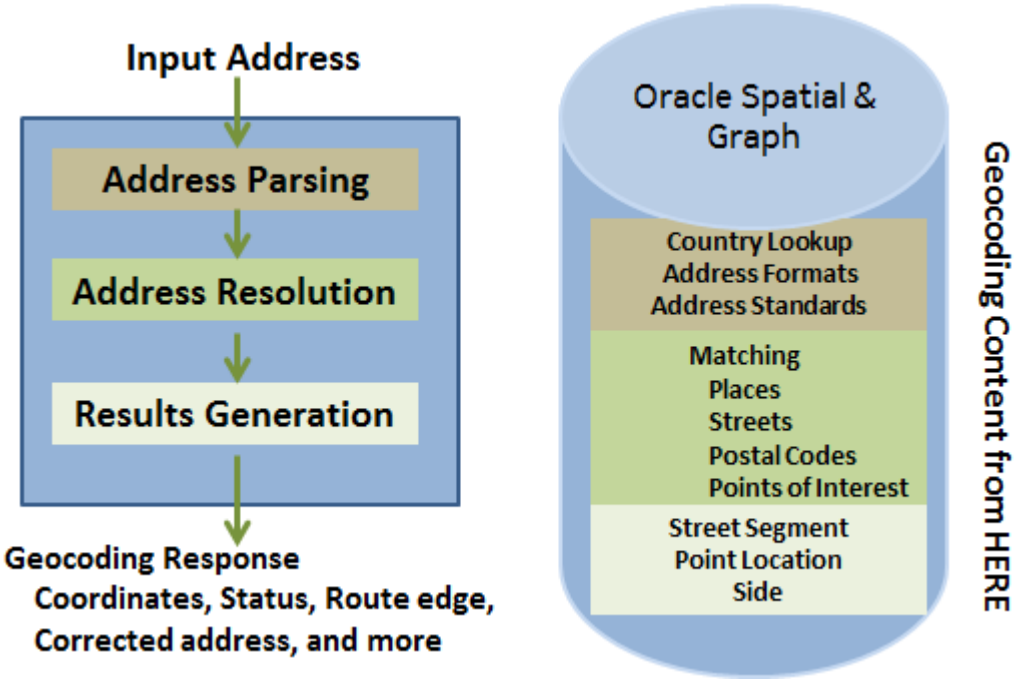
Oracle has two APIs for geocoding.

The SQL API is used within the database and is appropriate for geocoding and storing the returned location information directly in the database. The returned location can then be used for location analysis and display on a map.

The other API is an XML API for web-based geocoding (geocoding as a service). This API is typically used for web applications that are either dynamically displaying geocoded point locations on a map or generating routes, and for batch geocoding. The architecture of the web geocoding service is exactly the same as the MapViewer architecture (deployed as part of Oracle Fusion Miiddleware).



All geocoding requests are serviced in the Oracle database. The basic flow for geocoding requests is shown below:



Note that since the address parsing is driven off of geocoding metadata, there are unique advantages to the Oracle Spatial and Graph geocoder. With most off-the-shelf geocoder implementations, if the address parser can't resolve the input address appropriately your only option is to report a bug to your geocoder provider and hope at some time in the future they will accept the input address. With the Oracle Spatial and Graph geocoder customers have the ability to change the parser profile metadata to accept the altered address string. Then the altered parser profile can be fed back to the content provider, or the altered profile can become a strategic and competitive advantage related to your implementation.

SDO_GEO_ADDR Type and Attributes

All SQL geocoding functions are in the SDO_GCDR package. Looking in detail at one of the functions:

```
SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',SDO_GEO_ADDR(
  'US','DEFAULT', '15 Cosmo Place', 'San Francisco',null,
  'CA', '94109'))
FROM DUAL;
```

The geocoding function used above is `SDO_GCDR.GEOCODE_ADDR`. This signature of the `GEOCODE_ADDR` function requires two arguments:

The first argument, HERE_SF, is the owner into which geocoding data was loaded. If the user executing the geocoder request is not the owner of the data, then the owner will have to grant select privileges on all of the Geocoding tables (tables associated with the geocoder all have names beginning with 'GC') to the user executing the request (or to a role associated with the geocoding user).

The second argument is the SDO_GEO_ADDR type. The SDO_GEO_ADDR type has several constructors in Oracle. The constructor used in this version of the function is:

ATTRIBUTE	TYPE	Description
COUNTRY	VARCHAR2	Two character country code
MATCHMODE	VARCHAR2	Match description required
STREET	VARCHAR2	Number and Street
SETTLEMENT	VARCHAR2	Town or city
MUNICIPALITY	VARCHAR2	Not usually used in the US (County)
REGION	VARCHAR2	State or State Abbreviation
POSTALCODE	VARCHAR2	Zip or postal code

The two character country code shown (US) is that for the United States. Two digit character codes can be looked up in the table GC_M_ISO_COUNTRY_CODES, matching the COUNTRY_NAME with the COUNTRY_CODE_2 column.

The MATCHMODE is the requested accuracy of the match.

EXACT	Requires all attributes of the input address match (number, street, suffix, town, postal code). If all attributes do not match, the match drops back to postal code, city or town, or state
RELAX_STREET_TYPE	Allows match if the street type does not match. If the user inputs Oak Lane, and Oak Street is found then the match is allowed.
RELAX_POI_NAME	If a point of interest imperfectly matches, allow the match to proceed if there are no other issues such as name conflicts.
RELAX_HOUSE_NUMBER	Allows house number and street type to vary from input geocoded request. If street name doesn't match, falls back to postal code, city or town, or, as a last resort, state.
RELAX_BASE_NAME	Allows fuzzy match based on improper street name input. If the user inputs Maine Lane and only Main Street matches, a result based on street match is returned.
RELAX_POSTAL_CODE	Allows postal code, base name, house number, and street type to be inexact for geocoding. This is the default for Oracle 11g.
RELAX_BUILTUP_AREA	The same as relax postal code, plus the address doesn't have to be in the same city/town as long as it is in the same county.
RELAX_ALL	Same as RELAX_BUILTUP_AREA.
DEFAULT	Same as RELAX_POSTAL_CODE. In Oracle 10g, the same as RELAX_BASE_NAME.

Now that we understand the input format for this geocoding request, we can look at the returned values. Most geocoding requests return either an SDO_GEO_ADDR object or a VARRAY (varying length array) of SDO_GEO_ADDR objects.

The returned SDO_GEO_ADDR object looks very different from the input that we saw to the function. Let's look at that function again, along with the returned results:

```
SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',SDO_GEO_ADDR(
```

```
  'US','DEFAULT', '15 Cosmo Place', 'San Francisco',null,
  'CA', '94109'))
FROM DUAL;
```

The returned SDO_GEO_ADDR object is as follows:

```
SDO_GEO_ADDR(0, SDO_KEYWORDARRAY(), NULL, 'COSMO PL', NULL,
NULL,'SAN FRANCISCO', 'SAN FRANCISCO', 'CA', 'US', '94109',
NULL,'94109', NULL, '15', 'COSMO', 'PL', 'F', 'F', NULL,
NULL, 'R', .222222222, 199220154, '????#ENUT?B281CP?', 1,
'DEFAULT', -122.41225, 37.7880956, '???10101014??000?')
```

The attributes of the returned SDO_GEO_ADDR object are as follows:

ID - Unused

ADDRESSLINES – A free format way of specifying input to the geocoder. This field is empty in a geocode response.

PLACENAME – Specifies a point of interest (POI) name. For example, San Diego Zoo

STREETNAME – Specifies the street name, including the street type. For example, University Avenue

INTERSECTSTREET – Used as input when doing cross street (intersection) geocoding.

SECUNIT – Specifies the secondary unit, for instance an apartment number or building number.

SETTLEMENT – Specifies the lowest-level administrative area for the address. In the US, this is city or town.

MUNICIPALITY – This is the next highest administrative area after settlement. In the US, this is the county.

REGION – This is the next highest administrative area after municipality (when used), or above settlement if municipality is unused. In the US, this is state.

COUNTRY – Specifies either the Country name or two character ISO country code.

POSTALCODE – Specifies the postal code. In the United States, the postal code is the 5-digit ZIP Code.

POSTALADDONCODE – Specifies the optional part of the postal code. In the United States, the postal add-on code is the last four digits of a ZIP code ZIP+4 (5-4) format.

FULLPOSTALCODE – Specifies the full postal code, including the postal code and postal add-on code. In the US, this is ZIP + 4.

POBOX – Specified the post office box number.

HOUSENUMBER – Specifies the house number. For example, 75 in 75 Elm Street

BASENAME – Specifies the base name of the street. For example, ELM in 75 Elm Street

STREETTYPE – Specifies the type of the street. For example, ST in 75 Elm Street

STREETTYPEBEFORE – Unused

STREETTYPEATTACHED – Unused

STREETPREFIX – Specifies the prefix for the street. For example, S in 75 South Elm Street

STREETSUFFIX – Specifies the suffix for the street. For example: NW in 75 Elm Street Northwest

SIDE – Specifies the side of the street (L for left, R for right) that the house is on. This is determined by the direction the road feature is defined in the database (start node to end node).

PERCENT – Specifies the percent of the distance along the road segment to travel to the destination. Specified as a value between 0 and 1, this number is multiplied by 100 to get the actual percent. The percentage is determined along the direction the road feature is defined in the database.

EDGEID – Specifies the edge ID of the road segment returned in this geocode request.

ERRORMESSAGE – Specifies a string associated with each element of the geocoding operation (e.g. HOUSENUMBER, BASENAME, STREETTYPE, etc.). Oracle recommends using the MATCHVECTOR attribute of the SDO_GEO_ADDR type. For a full explanation, see the Oracle Spatial and Graph Developer's Guide.

MATCHCODE – Specifies the accuracy of the results of a geocode operation as a number. The numbers and their meanings are:

1 – Exact match. All fields in the input geocode operation matched values in the geocoding data set.

2 – All of the input fields of the geocoding operation match the geocoding data except the street type, prefix, or suffix

3 – All of the input fields of the geocoding operation match except the house or building number. Also the street type, prefix, or suffix may not match as well.

4 – The address does not match, but the city name and postal code do match.

10 – The postal code does not match the input geocoding request, but the city name does match.

11 – The postal code matches the data used for geocoding, but the city name does not match.

MATCHMODE - Match mode as specified as input to the geocoding operation, as specified above (EXACT, RELAX_STREET_TYPE, etc).

LONGITUDE – The Longitude coordinate value. This is half of the Longitude/Latitude pair that represents a location on the Earth.

LATITUDE – The Latitude coordinate value. This is the other part of the coordinate that represents an Earth location.

MATCHVECTOR – Specifies how accurate the geocode response is. Each element of the string corresponds to an element in the geocode request. Oracle recommends using the MATCHVECTOR to understand details about the geocoding request. For more information, see the Oracle Spatial and Graph Developer's Guide.

As previously mentioned, geocoding is often used in the context of routing applications. Note elements in the geocode response such as SIDE, PERCENT, and EDGEID are all associated with routing applications that use these components when generating a route from one geocoded location to another geocoded location. Also note that *routing data* is different content than geocoding data, although the routing data has to be associated with geocoding data (so the returned EDGEID matches in the geocode data and the routing data).

The SDO_GCDR.GEOCODE_ADDR Function

Oracle supplies several different geocoding functions. Let's look at the most common geocoding function, and we will see how it can be applied in several different ways.

Street Address Geocoding

The first example, as shown previously, allows us to input an address and return the location:

```
SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',SDO_GEO_ADDR(
  'US','DEFAULT', '15 Cosmo Place', 'San Francisco',null,
  'CA', '94109'))
FROM DUAL;

SDO_GEO_ADDR(0, SDO_KEYWORDARRAY(), NULL, 'COSMO PL', NULL,
NULL,'SAN FRANCISCO', 'SAN FRANCISCO', 'CA', 'US', '94109',
NULL,'94109', NULL, '15', 'COSMO', 'PL', 'F', 'F', NULL,
NULL, 'R', .222222222, 199220154, '????#ENUT?B281CP?', 1,
'DEFAULT', -122.41225, 37.7880956, '???10101014??000?')
```

Since we are geocoding in order to place locations on a map or do location analytics, we can use our knowledge of geocoding results to focus in on the parts that are most important to us.

If we think of all of the attributes in the results of a geocoder operation, there are some things that seem very important for display on a map. Minimally, in order to display the results on a map, we will need to capture the longitude and latitude of the geocoded location. We can return the longitude and latitude only by modifying our geocoding query:

```
SELECT GC.RESULT.LONGITUDE, GC.RESULT.LATITUDE
FROM (SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',
        SDO_GEO_ADDR( 'US', 'DEFAULT', '15 Cosmo Place',
        'San Francisco', null, 'CA', '94109')) as result
      FROM DUAL) GC;

RESULT.LONGITUDE RESULT.LATITUDE
```

```
---------------- ----------------
      -122.41225       37.7880956
```

That seems like enough to be able to display a location on a map, and it is. But we might care to know how successful the geocode operation was so we can conditionally decide whether to store the results. To do this, we can add another attribute to the select list:

```
SELECT GC.RESULT.LONGITUDE, GC.RESULT.LATITUDE,
   GC.RESULT.MATCHCODE
FROM (SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',
        SDO_GEO_ADDR( 'US', 'DEFAULT', '15 Cosmo Place',
        'San Francisco', null, 'CA', '94109')) as result
      FROM DUAL) GC;

RESULT.LONGITUDE RESULT.LATITUDE RESULT.MATCHCODE
---------------- --------------- ----------------
       -117.1394       32.7484441                1
```

HERE Point Addressing in Oracle

Oracle 11.2 and higher includes the ability to geocode to a predefined point location using a new table, GC_ADDRESS_POINT_NVT. The point location is related to the location of the parcel/land plot along a road segment, and is the state of the art with respect to returning highly accurate geocoding results. More accurate geocodes translate into more accurate analytics and mobile workforce optimization. This point addressing capability is enabled by premium point geocoding content from HERE.

A user can determine if a point geocode was returned by looking at the MATCHVECTOR attribute of the geocoding result. The third place within the match vector will be a "?" if the point address table doesn't exist, or if a match for the house number along the segment was not found. The value will be "0" if the match was found. For example:

```
SELECT GC.RESULT.LONGITUDE, GC.RESULT.LATITUDE,
   GC.RESULT.MATCHCODE, SUBSTR(GC.RESULT.MATCHVECTOR,3,1)
FROM (SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',
        SDO_GEO_ADDR( 'US', 'DEFAULT', '3832 Market St',
        'San Francisco', null, 'CA', '94131')) as result
      FROM DUAL) GC;

RESULT.LONGITUDE RESULT.LATITUDE RESULT.MATCHCODE SUBS
---------------- --------------- ---------------- ----
       -122.4426        37.75317                1 0
```

The "0" returned above shows a point address match.

Also note that by default the point returned by Oracle falls along a street segment. If you want to return the parcel centroid, you can change the point that is returned by altering the table:

```
alter table gc_address_point_nvt rename column addr_long to addr_long_centerline;
alter table gc_address_point_nvt rename column addr_lat to addr_lat_centerline;
alter table gc_address_point_nvt rename column display_long to addr_long;
alter table gc_address_point_nvt rename column display_lat to addr_lat;
```

If desired, you cango back to the default behavior by renaming the columns back to their default values.

Cross Street (Intersection) Geocoding

In the Oracle Geocoder, you can specify an intersection to geocode to. For instance, you can specify the "long form" of the `SDO_GEO_ADDR` type as input to the geocoding operation (the long form, as described above, includes an `INTERSECTSTREET` attribute) and return the location of the intersection.

To create the input to this function requires a really long constructor due to the number of attributes in the `SDO_GEO_ADDR` type:

```
SELECT GC.RESULT.LONGITUDE, GC.RESULT.LATITUDE,
  GC.RESULT.MATCHCODE
FROM (SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',
        SDO_GEO_ADDR(1, NULL, NULL,'CLAY STREET',
          'Powell St.', NULL, 'SAN FRANCISCO', NULL, 'CA',
          'US', NULL, NULL, NULL, NULL, NULL, NULL, NULL,
           NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
           NULL, 'DEFAULT', NULL, NULL)) RESULT
       FROM DUAL) GC;

RESULT.LONGITUDE RESULT.LATITUDE RESULT.MATCHCODE
---------------- --------------- ----------------
      -122.40953        37.79388                1
```

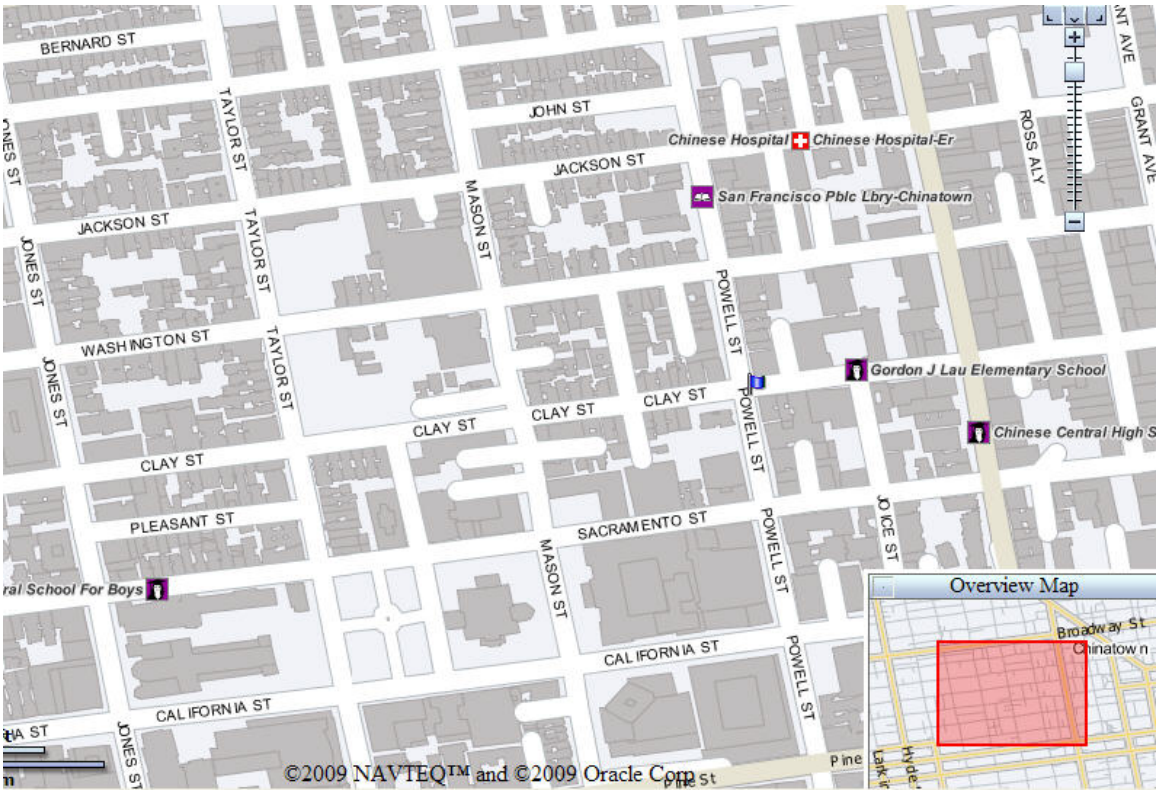
The call above works fine, but it is a little difficult to follow due to the large number of attributes. We can write a function to help us include only the attributes we need for this operation in the SDO_GEO_ADDR type:

```
CREATE OR REPLACE FUNCTION GEO_ADDR_FROM_INTERSECTION(
  STREET1    VARCHAR2,
  STREET2    VARCHAR2,
  SETTLEMENT VARCHAR2,
  REGION     VARCHAR2,
  COUNTRY    VARCHAR2,
  MATCHMODE  VARCHAR2)
  RETURN SDO_GEO_ADDR
AS
  GEO_ADDR SDO_GEO_ADDR ;
BEGIN
  GEO_ADDR := SDO_GEO_ADDR() ;
  GEO_ADDR.STREETNAME := STREET1 ;
  GEO_ADDR.INTERSECTSTREET := STREET2 ;
  GEO_ADDR.SETTLEMENT := SETTLEMENT ;
  GEO_ADDR.COUNTRY := REGION ;
  GEO_ADDR.COUNTRY := COUNTRY ;
  GEO_ADDR.MATCHMODE := MATCHMODE ;
  RETURN GEO_ADDR ;
```

```
END;
/
```

We can use this function in our geocoder call:

```
SELECT GC.RESULT.LONGITUDE, GC.RESULT.LATITUDE, GC.RESULT.MATCHCODE
FROM (SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',
        GEO_ADDR_FROM_INTERSECTION(
          'CLAY STREET', 'Powell St.', 'SAN FRANCISCO',
          'CA','US', 'EXACT')) RESULT
      FROM DUAL) GC;

RESULT.LONGITUDE RESULT.LATITUDE RESULT.MATCHCODE
---------------- --------------- ----------------
      -122.40953        37.79388                1
```



POI (Point of Interest) Geocoding

The final geocoding operation we will look at in depth is POI, or Point of Interest Geocoding. This geocoding operation requires the same form of the SDO_GEO_ADDR type as the previous (cross street) geocoding operation, but in this operation we need the PLACENAME attribute.

Let's create a function to geocode from a point of interest:

```
CREATE OR REPLACE FUNCTION GEO_ADDR_FROM_POI(
  POINAME    VARCHAR2,
  SETTLEMENT  VARCHAR2,
  REGION      VARCHAR2,
  COUNTRY     VARCHAR2,
  MATCHMODE   VARCHAR2)
  RETURN SDO_GEO_ADDR
AS
  GEO_ADDR SDO_GEO_ADDR ;
BEGIN
  GEO_ADDR := SDO_GEO_ADDR() ;
  GEO_ADDR.PLACENAME := POINAME ;
  GEO_ADDR.SETTLEMENT := SETTLEMENT ;
  GEO_ADDR.COUNTRY := REGION ;
  GEO_ADDR.COUNTRY := COUNTRY ;
  GEO_ADDR.MATCHMODE := MATCHMODE ;
  RETURN GEO_ADDR ;
END;
/
```

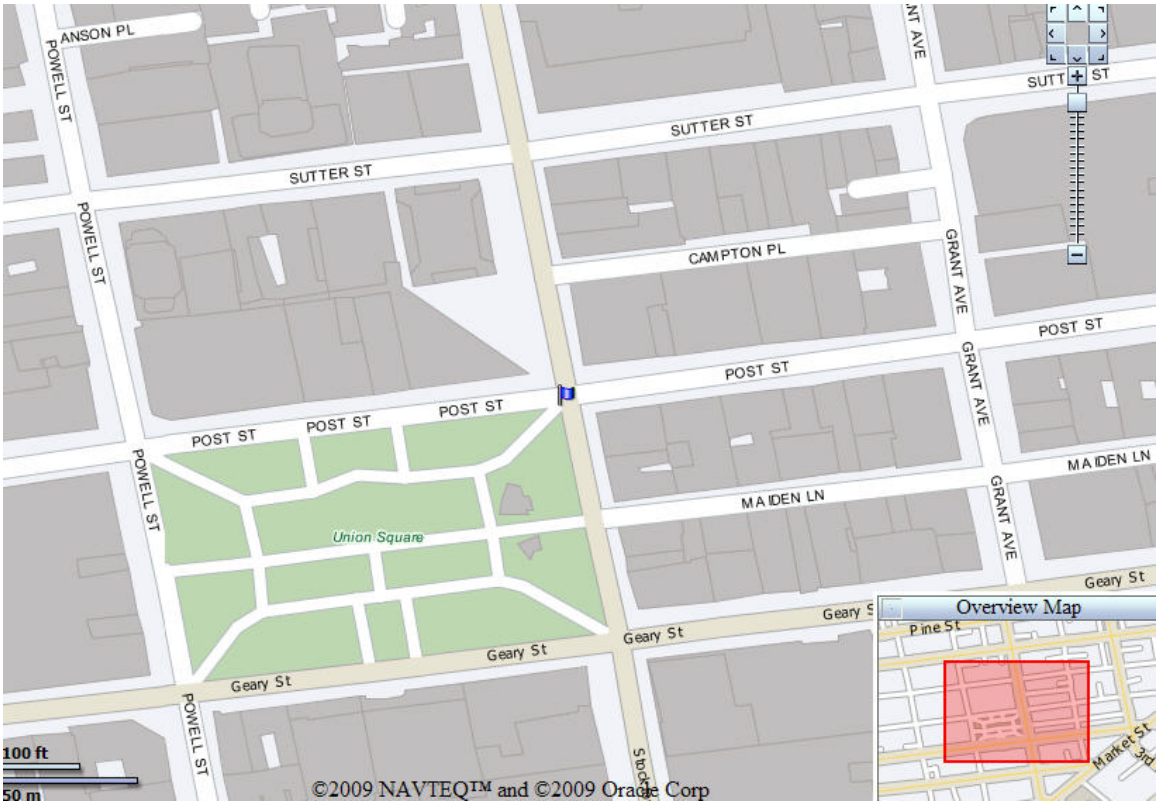
We can use this function in our geocoder call:

```
SELECT GC.RESULT.LONGITUDE, GC.RESULT.LATITUDE,
   GC.RESULT.MATCHCODE
FROM (SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',
        GEO_ADDR_FROM_POI( 'UNION SQUARE',
        'SAN FRANCISCO', 'CA','US', 'EXACT')) RESULT
      FROM DUAL) GC;

RESULT.LONGITUDE RESULT.LATITUDE RESULT.MATCHCODE
---------------- --------------- ----------------
      -122.40684        37.78845                1
```

Note also the extensive search capabilities Oracle provides. The POI table contains over two million points of interest. We can search directly on a POI name, and if it is uniquely named in our database we can find it without including the city and state:

```
SELECT GC.RESULT.LONGITUDE, GC.RESULT.LATITUDE,
```

```
GC.RESULT.MATCHCODE
FROM (SELECT SDO_GCDR.GEOCODE_ADDR('HERE_SF',
GEO_ADDR_FROM_POI( 'UNION SQUARE',
NULL, NULL,'US', 'EXACT')) RESULT
FROM DUAL) GC;

RESULT.LONGITUDE RESULT.LATITUDE RESULT.MATCHCODE
---------------- --------------- ----------------
      -122.40684        37.78845                1
```



Other Geocoding Functions

There are other geocoding functions in Oracle that are useful that we will not include in detail. However, it may be important for you to know about these capabilities, so we will briefly introduce each one.

`SDO_GCDR.REVERSE_GEOCODE` – This function, as the name implies, does the reverse of a geocode operation. It takes as input the longitude and latitude coordinates, and returns the closest address.

`SDO_GCDR.GEOCODE_ADDR_ALL` – This function has the same parameters as SDO_GCDR.GEOCODE_ADDR, but adds in a MAX_RES_NUM parameter. The function returns up to MAX_RES_NUM results. This is used when a geocoding request could result in many matches. For example, in a web application if a user is looking for a particular

address but it is specified incompletely the application can be present them with a list, and the user chooses the most appropriate store.

`SDO_GCDR.GEOCODE` – This function takes a free format address specified as a set of keywords in an `SDO_KEYWORD_ARRAY`, and returns an `SDO_GEO_ADDR` object.

`SDO_GCDR.GEOCODE_AS_GEOMETRY` – This function takes a free format address as input, and returns a value as `SDO_GEOMETRY` as a result. There is no way to determine the accuracy of the geocode using this function.

`SDO_GCDR.BATCH_GEOCODE` – This function takes a set of addresses as an input, and returns the geocoded set of address objects in the geocode response. The function is most useful when the geocoder is used as a web service, as it reduces the number of round trips to the geocoder.

Map Your Geocoding Results

One of the most common ways to show customer locations is to geocode the addresses, store them in the Oracle database, and then index the locations for fast retrieval.

Storing Geocoding Results in Oracle

Result of geocoding operations can be stored in Oracle using **SDO_GEOMETRY** (the Oracle type for storing location information). Let's look at how we might add a column of type **SDO_GEOMETRY** to a CUSTOMERS table, geocode the address information in the table and store the results.

Let's start with a CUSTOMERS table with some data:

```
CREATE TABLE CUSTOMERS
 (CUSTOMER_ID NUMBER NOT NULL UNIQUE,
  LAST_NAME VARCHAR2(30),
  FIRST_NAME VARCHAR2(20),
  ADDRESS   VARCHAR2(70),
  CITY  VARCHAR2(40),
  STATE_ABBRV VARCHAR2(2),
  POSTAL_CODE VARCHAR2(10));

insert into customers values (1,'Doe','Jane',
  '460 EDDY ST','SAN FRANCISCO','CA','94109');
insert into customers values (2,'Roy','James',
  '111 CAMBON DR','SAN FRANCISCO','CA','94132');
insert into customers values (3,'Pear','Fiona',
  '570 INDIANA ST','SAN FRANCISCO','CA','94107');
insert into customers values (4,'Taylor','Andrea',
  '2099 FOLSOM ST','SAN FRANCISCO','CA','94110');
insert into customers values (5,'Buckley','Michael',
  ' 68 LELAND AVE','SAN FRANCISCO','CA','94134');
insert into customers values (6,'Farr','Brian',
  ' 1975 POST ST','SAN FRANCISCO','CA','94115');
insert into customers values (7,'King','Doris',
  ' 20 HERON ST','SAN FRANCISCO','CA','94103');
insert into customers values (8,'Hoople','Motthe',
  ' 1496 MARKET ST','SAN FRANCISCO','CA','94102');
insert into customers values (9,'Orafice','Helene',
  ' 559 PACIFIC AVE','SAN FRANCISCO','CA','94133');
insert into customers values (10,'Scott','Joyce',
  ' 1519 POLK ST','NATIONAL CITY','CA','91950');
commit;
```

Now that we have a typical customers table let's location-enable it by adding the Oracle location data type, **SDO_GEOMETRY**:

```
ALTER TABLE CUSTOMERS ADD (LOCATION SDO_GEOMETRY);
```

Now we will create a function to update the location column in the customers table:

```
CREATE OR REPLACE FUNCTION GEOCODE_ADDR(
   STREET_ADDR VARCHAR2, CITY VARCHAR2,
   STATE VARCHAR2, ZIP VARCHAR2)
RETURN SDO_GEOMETRY
DETERMINISTIC AS
  GEO_ADDR SDO_GEO_ADDR;
BEGIN
  GEO_ADDR :=
   SDO_GCDR.GEOCODE_ADDR('HERE_SF',SDO_GEO_ADDR('US',
'DEFAULT',STREET_ADDR,CITY,NULL,STATE,ZIP));
  IF GEO_ADDR.MATCHCODE > 3
  THEN
     RETURN NULL;
  ELSE
     RETURN SDO_GEOMETRY(2001,8307,SDO_POINT_TYPE(GEO_ADDR.LONGITUDE,
GEO_ADDR.LATITUDE, NULL),NULL,NULL);
  END IF;
END;
/

UPDATE CUSTOMERS SET LOCATION =
  GEOCODE_ADDR(ADDRESS,CITY,STATE_ABBRV,POSTAL_CODE);
```

```
select last_name,location from customers;

LAST_NAME          LOCATION(SDO_GTYPE, SDO_SRID,
-----------------------------------------------------------
Doe
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.4152, 37.7836, NULL), NULL, NULL)

Roy
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.4742, 37.7170, NULL), NULL, NULL)

Pear
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.3916, 37.7631, NULL), NULL, NULL)

Taylor
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.4151, 37.7636, NULL), NULL, NULL)

Buckley
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.4051, 37.7117, NULL), NULL, NULL)

Farr
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.43428, 37.785, NULL), NULL, NULL)

King
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.4086, 37.7745, NULL), NULL, NULL)

Hoople
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.4192, 37.7752, NULL), NULL, NULL)

Orafice
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.4042, 37.7972, NULL), NULL, NULL)

Scott
```

Note the last customer, Scott, did not geocode correctly. A quick analysis determined that our San Francisco data set does not include geocoding data for National City, and the customer had given us the wrong city name, hence the geocode operation failed (returned a match code > 3).

Also note a trigger can be created so every time the customers table has a record inserted or updated, the geocode operation will recur automatically.

For example, thefollowing trigger will automatically update the location column:

```
CREATE OR REPLACE TRIGGER location_geocode
  BEFORE INSERT OR UPDATE OF address ON customers
FOR EACH ROW WHEN (new.location IS NULL)
BEGIN
  :new.location :=
      geocode_addr(:new.address, :new.city,
                   :new.state_abbrv, :new.postal_code);
END;
/
```

Now we can update the information for customer SCOTT, and the new value will be reflected in the location column:

```
update customers set address=
  ' 1519 POLK ST', city='San Francisco',
  state_abbrv='CA', postal_code='94109'
where last_name='Scott';

select location from customers where last_name='Scott';

LOCATION(SDO_GTYPE, SDO_SRID, SDO_POINT(X, Y, Z)
-----------------
SDO_GEOMETRY(2001, 8307, SDO_POINT_TYPE(-122.42071, 37.7907408, NULL), NULL, NULL)
```

Registering Location Data in Oracle

To prepare to see our data on a map, we need to register the location information (the **SDO_GEOMETRY** column) in Oracle. This is done by inserting the pertinent information into the **USER_SDO_GEOM_METADATA** view:

```
INSERT INTO USER_SDO_GEOM_METADATA VALUES ('CUSTOMERS','LOCATION',
  SDO_DIM_ARRAY(
     SDO_DIM_ELEMENT('LON',-180,180,0.5),
     SDO_DIM_ELEMENT('LAT',-90,90,0.5)),
  8307);
```

Indexing Location Data in Oracle

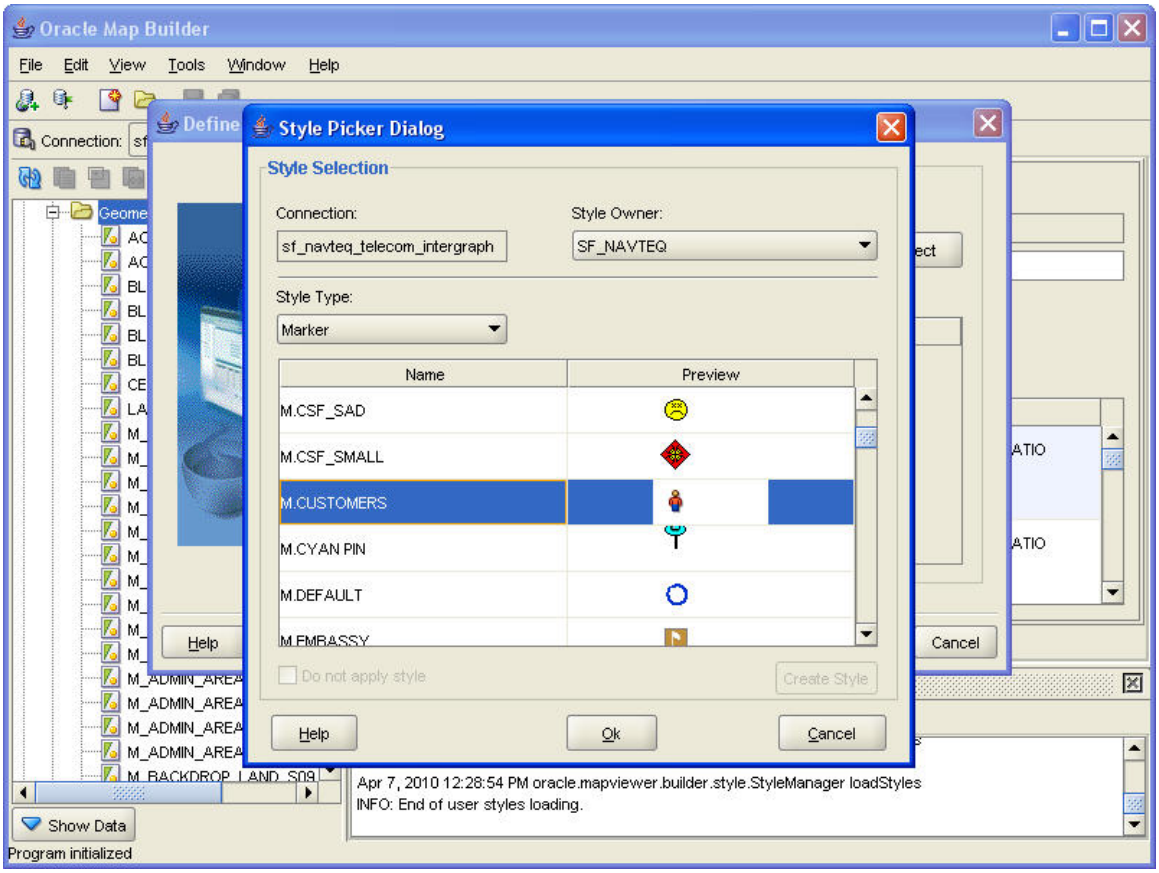
We build indexes on location data in Oracle for the same reason we build indexes on other columns we will be searching in Oracle. The indexes are built for very fast access to specific data we care about. In this case, the index is based on spatial relationships, and lets us search based on proximity.

```
CREATE INDEX CUSTOMERS_SIDX ON CUSTOMERS(LOCATION)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX PARAMETERS
  ('LAYER_GTYPE=POINT');
```

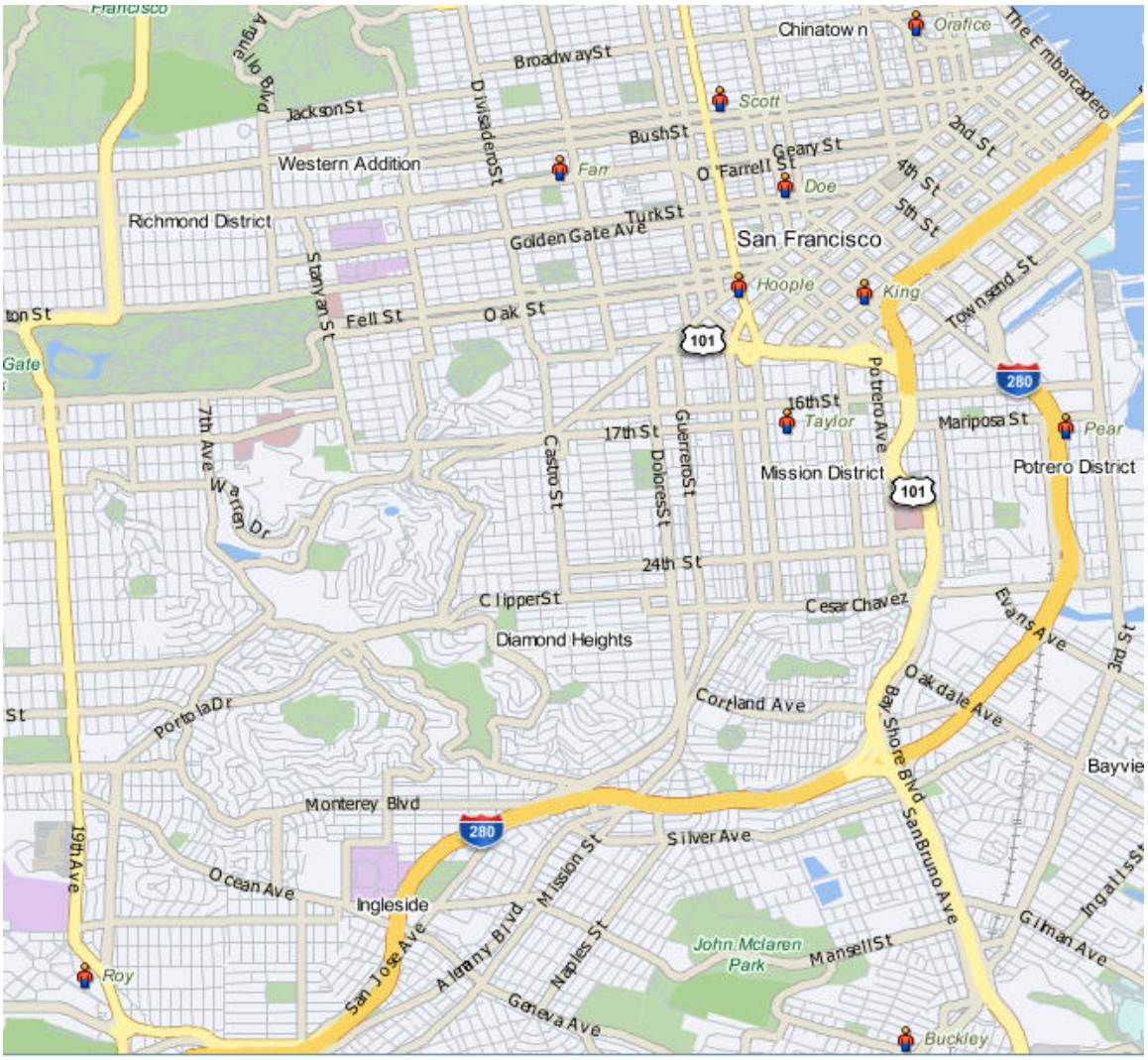
Putting Your Customers on the Map

Oracle Fusion Middleware allows us to display locations and do business analysis based on location. It is easy to add a new set of information by creating either a theme in MapViewer or a Feature of Interest layer in Oracle Maps.

We can add customers to our maps by creating a theme (the location column in the CUSTOMERS tables with the marker and the label that will be used for display):



Finally, the theme is added to a map for display:



About HERE

HERE, a Nokia business, offers a HERE Map Content and the HERE Platform, a cloud-based location platform to enterprises and government users around the globe. HERE is the Oracle-preferred supplier of geographic content to Oracle Applications. To learn more about HERE, visit http://here.com/business/enterprise.