Oracle Spatial 11*g*
Geocoder

An Oracle Technical White Paper
July 2011

Contents

1 Introduction

Geocoding is a procedure that uses postal addresses to derive geographic (longitude, latitude) locations. The Oracle Geocoder provides geocoding and reverse-geocoding services, and performs point-of-interest (POI) matching that returns complete addresses for points of interest.

The Oracle Geocoder is a geocoding engine implemented as Java-stored procedures inside an Oracle database server. In addition to geocoding PL/SQL APIs, client-side Java APIs are provided to facilitate the integration of the server-side geocoder with client-side Java applications.

The Oracle Geocoder is supported in three forms: the database geocoder, the J2EE geocoder and the Thin-Client geocoder. The database geocoder uses PL/SQL APIs to access address and coordinate information stored in database tables; the J2EE geocoder uses an XML API to provide a service that accesses the address and coordinate information stored in database tables; and the Thin-Client geocoder uses a client-side Java API to access address and coordinate information stored in database tables. The three geocoders all access the same underlying database tables. These tables and their indexes constitute the geocoder schema. The geocoder schema is typically supplied by a data provider and is generally ready to use out-of-the-box, however, the geocoder schema can also be built by a user.

This white paper describes the geocoder schema used by the Oracle Geocoder and how data is accessed and interpreted for geocoding. It also describes how the Oracle Geocoder is configured for use. The examples in this paper are based on the United States address-format conventions, unless stated otherwise.

The features of the Oracle Geocoder that are specific to Oracle Spatial 11*g* Release 2 are point-based geocoding, Oracle WebLogic Server support and the Thin-Client geocoder.

2 Geocoder Schema

The geocoder schema is made up of seven data tables and their indexes, and three profile tables. The data tables store address and coordinate information, and the profile tables store address-format information used by the geocoder to parse addresses entered by a user.

2.1 Data Tables and Indexes

The data tables, GC_ROAD_SEGMENT_<suffix>, GC_AREA_<suffix>, GC_ROAD_<suffix>, GC_INTERSECTION_<suffix>, GC_POSTAL_CODE_<suffix>, GC_POI_<suffix> and GC_ADDRESS_POINT_<suffix> store address and coordinate information for a single country or group of countries of interest. The mandatory suffix for these tables is data-provider- or user-specified and is typically used to identify the country or group of countries to which the data belongs, for example, “US” for the United States or “EU” for Europe. However, any 5-characters-or-less can be used for the table-name suffix and the same suffix must also be used in the creation of indexes on these tables. The seven tables and their indexes are described in this section:

2.1.1 GC_ROAD_SEGMENT Table and Indexes

The GC_ROAD_SEGMENT_<suffix> table (for example, GC_ROAD_SEGMENT_US) stores road-segment information for the country or group of countries associated with the table-name suffix. A road segment is the portion of a road between two continuous intersections along the road; an intersection occurs when roads meet or cross each other. A road segment can also be the portion of a road between the start (or end) of the road and its closest intersection along the road, or it can be the entire length of a road if there are no intersections along the road. The GC_ROAD_SEGMENT_<suffix> table contains one row for each road segment and has the columns shown in Table 1.

Table 1: GC_ROAD_SEGMENT_<suffix> Table

Column Name	Data Type	Description
ROAD_SEGMENT_ID	NUMBER	ID number of the road segment. (Required)
ROAD_ID	NUMBER	ID number of the road to which the road segment belongs. (Required)
L_ADDR_FORMAT	VARCHAR2(1)	Left-side address format. Specify N if there are house numbers on the left side of the road segment; leave null if there are no house numbers on the left side of the road segment. (Required)
R_ADDR_FORMAT	VARCHAR2(1)	Right-side address format. Specify N if there are house numbers on the right side of the road segment; leave null if there are no house numbers on the right side of the road segment. (Required)
L_ADDR_SCHEME	VARCHAR2(1)	Numbering scheme for house numbers on the left side of the road segment: O (all odd numbers), E (all even numbers), or M (mixture of odd and even numbers). (Required)
R_ADDR_SCHEME	VARCHAR2(1)	Numbering scheme for house numbers on the right side of the road segment: O (all odd numbers), E (all even numbers), or M (mixture of odd and even numbers). (Required)
START_HN	NUMBER(5)	The lowest house number on the road segment. (Required)
END_HN	NUMBER(5)	The highest house number on the road segment. (Required)
L_START_HN	NUMBER(5)	The leading numerical part of the left-side starting house number. (See the explanation of house numbers after this table.) (Required)
L_END_HN	NUMBER(5)	The leading numerical part of the left-side ending house number. (See the explanation of house numbers after this table.) (Required)

Column Name	Data Type	Description
R_START_HN	NUMBER(5)	The leading numerical part of the right-side starting house number. (See the explanation of house numbers after this table.) (Required)
R_END_HN	NUMBER(5)	The leading numerical part of the right-side ending house number. (See the explanation of house numbers after this table.) (Required)
POSTAL_CODE	VARCHAR2(16)	Postal code for the road segment. If the left side and right side of the road segment belong to two different postal codes, create two rows for the road segment with identical values in all columns except for POSTAL_CODE. (Required)
GEOMETRY	SDO_GEOMETRY	Spatial geometry object (containing shape points) that represents the road segment. (Required)
COUNTRY_CODE_2	VARCHAR2(2)	2-letter ISO country code of the country to which the road segment belongs. (Required)
PARTITION_ID	NUMBER	Partition key used for partitioning geocoder data by geographic boundaries. If the data is not partitioned, set this value to 1. (Required)
L_START_HN2	VARCHAR2(10)	The second part of the left-side starting house number. (See the explanation of house numbers after this table.) (Required if the left-side starting house number has a second part)
L_END_HN2	VARCHAR2(10)	The second part of the left-side ending house number. (See the explanation of house numbers after this table.) (Required if the left-side ending house number has a second part)
R_START_HN2	VARCHAR2(10)	The second part of the right-side starting house number. (See the explanation of house numbers after this table.) (Required if the right-side starting house number has a second part)
R_END_HN2	VARCHAR2(10)	The second part of the right-side ending house number. (See the explanation of house numbers after this table.) (Required if the right-side ending house number has a second part)

House-Number Attributes

A house number is a descriptive part of an address that helps identify the location of a establishment along a road segment. In the Oracle Geocoder, a house number is divided into two parts: a leading numerical part, and a second part containing the rest of the house number. The leading numerical part is the numerical part of the house number that starts from the beginning of the complete house number string and ends just before the first

non-numeric character (if present). If the house number contains non-numeric characters, the second part of the house number is the portion from the first non-numeric character through the last character of the string. An example house number of 123 will therefore have a leading numerical part of 123 and a second part of null; and a house number of 123A23 will have a leading numerical part of 123 and a second part of A23.

The starting house number is the house number at the start-point of a road segment; the start-point of the road segment is the first shape point of the road-segment's geometry. The ending house number is the house number at the end-point of a road segment; the end-point of the road segment is the last shape point of the road-segment's geometry. The left- and right-side starting house numbers need not be lower than the left- and right-side ending house numbers. The house-number attributes found in the data tables follow these conventions in locating establishments along road segments.

GC_ROAD_SEGMENT Indexes

There are two required indexes on the GC_ROAD_SEGMENT_<suffix> table. They are created as illustrated in Example 1.

Example 1: GC_ROAD_SEGMENT_<suffix> Indexes

```
CREATE INDEX idx_<suffix>_road_geom ON gc_road_segment_<suffix> (geometry)
INDEXTYPE IS mdsys.spatial_index;

CREATE INDEX idx_<suffix>_road_seg_rid ON gc_road_segment_<suffix> (road_id,
start_hn, end_hn);
```

2.1.2 GC_AREA Table and Indexes

The GC_AREA_<suffix> table (for example, CG_AREA_US) stores information on the administrative areas for the country or group of countries associated with the table-name suffix. Information may be stored for up to 7 different levels of administrative areas, though 4 or 5 levels are common in most countries, with the first level representing the largest administrative area and the seventh level representing the smallest administrative area. In the US administrative hierarchy, a level-1 area represents a country, a level-2 area represents a state, a level-3 area represents a county, and a level-4 area represents a city.

To populate the GC_AREA table, you must specify an area ID for each level in the administrative hierarchy to which the area belongs. An area with ADMIN_LEVEL=1, i.e., a country, will have a LEVEL1_AREA_ID associated with it, but no data (0) in its LEVEL2_ through LEVEL7_AREA_ID columns. However, an area with ADMIN_LEVEL=3, i.e., a county, will have a LEVEL3_AREA_ID for the county, a LEVEL2_AREA_ID for the state, and a LEVEL1_AREA_ID for the country to which it belongs. In this example, LEVEL4_ through LEVEL7_AREA_ID columns will have no data (0). It is important to note, that the AREA_ID value for an area is the same as the area's LEVEL(admin_level)_AREA_ID. Therefore, if an area has an ADMIN_LEVEL=2, its AREA_ID and its LEVEL2_AREA_ID are equivalent.

The GC_AREA_<suffix> table contains the columns shown in Table 2.

Table 2: GC_AREA_<suffix> Table

Column Name	Data Type	Description
AREA_ID	NUMBER(10)	ID number of the administrative area. (Required)
AREA_NAME	VARCHAR2(64)	Name of the administrative area. (Required)
LANG_CODE	VARCHAR2(3)	3-letter ISO national language code for the language associated with the administrative area. (Required)
ADMIN_LEVEL	NUMBER(1)	Administrative hierarchy level for the administrative area. 1 to 7 are valid entries for this column. (Required)
LEVEL1_AREA_ID	NUMBER(10)	AREA_ID of the level-1 area to which the administrative area belongs. In the administrative hierarchy, the level-1 area is the country. (Required)
LEVEL2_AREA_ID	NUMBER(10)	AREA_ID of the level-2 area to which the administrative area belongs, if applicable.
LEVEL3_AREA_ID	NUMBER(10)	AREA_ID of the level-3 area to which the administrative area belongs, if applicable.
LEVEL4_AREA_ID	NUMBER(10)	AREA_ID of the level-4 area to which the administrative area belongs, if applicable.
LEVEL5_AREA_ID	NUMBER(10)	AREA_ID of the level-5 area to which the administrative area belongs, if applicable.
LEVEL6_AREA_ID	NUMBER(10)	AREA_ID of the level-6 area to which the administrative area belongs, if applicable.
LEVEL7_AREA_ID	NUMBER(10)	AREA_ID of the level-7 area to which the administrative area belongs, if applicable.
CENTER_LONG	NUMBER	Longitude value of the center of the administrative area. The center (longitude, latitude) value is set to the start- or end-point of the closest road segment to the center, depending on which point is closer. Oracle recommends that the CENTER_LONG and CENTER_LAT be correctly set. If these values are not set, the longitude, latitude values of the geocoded result for an area will be (0,0).
CENTER_LAT	NUMBER	Latitude value of the center of the administrative area. (See the explanation for the CENTER_LONG column.)

Column Name	Data Type	Description
ROAD_SEGMENT_ID	NUMBER(10)	ID number of the road segment to which the administrative area's center is set. This value must be correctly set if the geocoder is intended to work with the Oracle Spatial routing engine; otherwise, it can be set to any non-zero value, but it cannot be null. (Required)
POSTAL_CODE	VARCHAR2(16)	Postal code for the center of the administrative area. Oracle recommends that this attribute be correctly set. If this value is null, the postal code attribute of the geocoded result of an area will be null.
COUNTRY_CODE_2	VARCHAR2(2)	2-letter ISO country code of the country to which the administrative area belongs. (Required)
PARTITION_ID	NUMBER	Partition key used for partitioning geocoder data by geographic boundaries. If the data is not partitioned, set this value to 1. (Required)
REAL_NAME	VARCHAR2(64)	The real name of the administrative area, as spelled using the local language. This column is useful for area names that are not in English. For example, the German name of the city MUNICH is MÜNCHEN. MÜNCHEN can be spelled as MUNCHEN, but its REAL_NAME value should be MÜNCHEN. In the GC_AREA table for Germany, areas with the name MÜNCHEN and MUNCHEN both refer to the same area and have the same real name MÜNCHEN. If an area name has only English characters, set the REAL_NAME to the same value as the AREA_NAME. (Required)
IS_ALIAS	VARCHAR2(1)	Contains 'T' if the area is an alias for an officially recognized administrative area, and 'F' if it is not. For example, Manhattan is not an officially recognized administrative area, but it is used to refer to a part of New York City. In this case, Manhattan's IS_ALIAS value is set to T. (Required)
NUM_STREETS	NUMBER	The number of road segments inside the administrative area.

GC_AREA Indexes

There are two required indexes on the GC_AREA_<suffix> table. They are created as illustrated in Example 2.

Example 2: GC_AREA_<suffix> Indexes

```
CREATE INDEX idx_<suffix>_area_name_id ON gc_area_<suffix> (country_code_2,
area_name, admin_level);
```

```
CREATE INDEX idx_<suffix>_area_id_name ON gc_area_<suffix> (area_id, area_name,
country_code_2);
```

2.1.3 GC_ROAD Table and Indexes

The GC_ROAD_<suffix> table (for example, GC_ROAD_US) stores road information for the country or group of countries associated with the table-name suffix. A road is a collection of road segments with the same name in the same settlement area. A settlement area is the smallest administrative area used in addressing; in the US it is a city. The GC_ROAD_<suffix> table contains at least one row for each road; it may contain multiple rows for a road when the road is associated with multiple settlements. The columns for this table are shown in Table 3.

Table 3: GC_ROAD_<suffix> Table

Column Name	Data Type	Description
ROAD_ID	NUMBER	ID number of the road. (Required)
SETTLEMENT_ID	NUMBER(10)	ID number of the settlement to which the road belongs. (Required if the road is associated with a settlement)
MUNICIPALITY_ID	NUMBER(10)	ID number of the municipality to which the road belongs. The municipality is one level above a settlement in the administrative hierarchy. (Required if the road is associated with a municipality)
PARENT_AREA_ID	NUMBER(10)	ID number of the parent area of the municipality to which the road belongs. The parent area is one level above the municipality in the administrative hierarchy. (Required if the road is associated with a parent area)
LANG_CODE	VARCHAR2(3)	3-letter ISO national language code for the language of the road name. (Required)
NAME	VARCHAR2(64)	Name of the road, including the type (if any), the prefix (if any), and the suffix (if any). For example, N Main St as NAME. (Required)
BASE_NAME	VARCHAR2(64)	Name of the road, excluding the type (if any), the prefix (if any), and the suffix (if any). For example, N Main St as NAME, with Main as BASE_NAME. (Required)
PREFIX	VARCHAR2(32)	Prefix of the road name. For example, N Main St as NAME, with N as PREFIX. (Required if the road name has a prefix)
SUFFIX	VARCHAR2(32)	Suffix of the road name. For example, Main St NW as NAME, with NW as SUFFIX. (Required if the road name has a suffix)

Column Name	Data Type	Description
STYPE_BEFORE	VARCHAR2(32)	Street type that precedes the base name. For example, Avenue Victor Hugo as NAME, with Avenue as STYPE_BEFORE and Victor Hugo as BASE_NAME. (Required if the road type precedes the base name)
STYPE_AFTER	VARCHAR2(32)	Street type that follows the base name. For example, Main St as NAME, with St as STYPE_AFTER and Main as BASE_NAME. (Required if the road type follows the base name)
STYPE_ATTACHED	VARCHAR2(1)	Contains T if the street type is in the same word with the street name; contains F if the street type is a separate word from the street name. For example, in a German street address of 123 Beethovenstrasse, the street type is strasse, and it is in the same word with the street name, which is Beethoven. (Required)
START_HN	NUMBER(5)	The lowest house number on the road. It is returned when a specified house number is lower than this value.
CENTER_HN	NUMBER(5)	Leading numerical part of the center house number. The center house number is the house number at the start-point of the center road segment, which is located in the center of the entire road. (See the explanation of house-number attributes after Table 1.) It is returned when no house number is specified in an input address. (Required if there are houses on the road)
END_HN	NUMBER(5)	The highest house number on the road. It is returned when a specified house number is higher than this value.
START_HN_SIDE	VARCHAR2(1)	Side of the road of the lowest house number: L for left or R for right.
CENTER_HN_SIDE	VARCHAR2(1)	Side of the road of the center house number: L for left or R for right. The center house number is the house number at the start-point of the center road segment, which is located in the center of the entire road. (See the explanation of house-number attributes after Table 1.) (Required if there are houses on the road)
END_HN_SIDE	VARCHAR2(1)	Side of the road of the highest house number: L for left or R for right.
START_LONG	NUMBER	Longitude value of the lowest house number.
START_LAT	NUMBER	Latitude value of the lowest house number.

Column Name	Data Type	Description
CENTER_LONG	NUMBER	Longitude value of the center house number. The center house number is the house number at the start-point of the center road segment, which is located in the center of the entire road. (See the explanation of house-number attributes after Table 1.) (Required)
CENTER_LAT	NUMBER	Latitude value of the center house number. (See the explanation for the CENTER_LONG column.) (Required)
END_LONG	NUMBER	Longitude value of the highest house number.
END_LAT	NUMBER	Latitude value of the highest house number.
START_ROAD_SEG_ID	NUMBER(5)	ID number of the road segment at the start of the road.
CENTER_ROAD_SEG_ID	NUMBER(5)	ID number of the road segment at the center point of the road. (Required)
END_ROAD_SEG_ID	NUMBER(5)	ID number of the road segment at the end of the road.
POSTAL_CODE	VARCHAR2(16)	Postal code for the road. (Required)
COUNTRY_CODE_2	VARCHAR2(2)	2-letter ISO country code for the country to which the road belongs. (Required)
PARTITION_ID	NUMBER	Partition key used for partitioning geocoder data by geographic boundaries. If the data is not partitioned, set this value to 1. (Required)
CENTER_HN2	VARCHAR2(10)	The second part of the center house number. (See the explanation of house-number attributes after Table 1) (Required)

GC_ROAD Indexes

There are seven required indexes on the GC_ROAD_<suffix> table. They are created as illustrated in Example 3.

Example 3: GC_ROAD_<suffix> Indexes

```
CREATE INDEX idx_<suffix>_road_id ON gc_road_<suffix> (road_id);

CREATE INDEX idx_<suffix>_road_setbn ON gc_road_<suffix> (settlement_id,
base_name);

CREATE INDEX idx_<suffix>_road_munbn ON gc_road_<suffix> (municipality_id,
base_name);

CREATE INDEX idx_<suffix>_road_parbn ON gc_road_<suffix> (parent_area_id,
country_code_2, base_name);

CREATE INDEX idx_<suffix>_road_setbnsd ON gc_road_<suffix> (settlement_id,
soundex(base_name));
```

```
CREATE INDEX idx_<suffix>_road_munbnsd ON gc_road_<suffix> (municipality_id,
soundex(base_name));

CREATE INDEX idx_<suffix>_road_parbnsd ON gc_road_<suffix> (parent_area_id,
country_code_2, soundex(base_name));
```

2.1.4 GC_INTERSECTION Table and Index

The GC_INTERSECTION_<suffix> table (for example, GC_INTERSECTION_US) stores information on road intersections for the country or group of countries associated with the table-name suffix. An intersection occurs when roads meet or cross each other. The GC_INTERSECTION_<suffix> table contains the columns shown in Table 4.

Table 4: GC_INTERSECTION_<suffix> Table

Column Name	Data Type	Description
ROAD_ID_1	NUMBER	ID number of the first road on which the intersection is located. (Required)
ROAD_SEGMENT_ID_1	NUMBER	ID number of the road segment on the first road on which the intersection is located. (Required)
ROAD_ID_2	NUMBER	ID number of the second road on which the intersection is located. (Required)
ROAD_SEGMENT_ID_2	NUMBER	ID number of the road segment on the second road on which the intersection is located. (Required)
INTS_LONG	NUMBER	Longitude value of the intersection. (Required)
INTS_LAT	NUMBER	Latitude value of the intersection. (Required)
HOUSE_NUMBER	NUMBER	The leading numerical part of the house number at the intersection. If no left-side house number is available, choose the right-side house number. (Required)
HOUSE_NUMBER_2	VARCHAR2(10)	The second part of the house number at the intersection. (See the explanation of house-number attributes after Table 1.) (Required)
SIDE	VARCHAR2(1)	Side of the road on which the house at the intersection is located. Possible values: L (left) or R (right). (Required)
COUNTRY_CODE_2	VARCHAR2(2)	2-letter ISO country code of the country to which the house at the intersection belongs. (Required)
PARTITION_ID	NUMBER	Partition key used for partitioning geocoder data by geographic boundaries. If the data is not partitioned, set this value to 1. (Required)

GC_INTERSECTION Index

There is one required index on the GC_INTERSECTION_<suffix> table. It is created as illustrated in Example 4.

Example 4: GC_INTERSECTION_<suffix> Index

```
CREATE INDEX idx_<suffix>_inters ON gc_intersection_<suffix> (country_code_2,
road_id_1, road_id_2);
```

2.1.5 GC_POSTAL_CODE Table and Index

The GC_POSTAL_CODE_<suffix> table (for example, GC_POSTAL_CODE_US) stores postal code information for the country or group of countries associated with the table-name suffix, providing postal codes are used in the address format. This table contains one or more rows for each postal code; it may contain multiple rows for a postal code when the postal code is associated with multiple settlements. The GC_POSTAL_CODE_<suffix> table contains the columns shown in Table 5.

Table 5: GC_POSTAL_CODE_<suffix> Table

Column Name	Data Type	Description
POSTAL_CODE	VARCHAR2(16)	Postal code for the postal-code area. (Required)
SETTLEMENT_NAME	VARCHAR2(64)	Name of the settlement to which the postal code belongs. (Required if the postal code is associated with a settlement)
MUNICIPALITY_NAME	VARCHAR2(64)	Name of the municipality to which the postal code belongs. (Required if the postal code is associated with a municipality)
REGION_NAME	VARCHAR2(64)	Name of the region to which the postal code belongs. The region is the administrative area above the municipality used in addressing. (Required if the postal code is associated with a region)
LANG_CODE	VARCHAR2(3)	3-letter ISO national language code for the language associated with the postal-code area. (Required)
SETTLEMENT_ID	NUMBER(10)	ID number of the settlement to which the postal code belongs. (Required if the postal code is associated with a settlement)
MUNICIPALITY_ID	NUMBER(10)	ID number of the municipality to which the postal code belongs. (Required if the postal code is associated with a municipality)
REGION_ID	NUMBER(10)	ID number of the region to which the postal code belongs. (Required if the postal code is associated with a region)

Column Name	Data Type	Description
CENTER_LONG	NUMBER	Longitude value of the center of the postal-code area. The center (longitude, latitude) value is set to the start- or end-point of the closest road segment to the center, depending on which point is closer. Oracle recommends that the CENTER_LONG and CENTER_LAT be correctly set. If these values are not set, the longitude, latitude values of the geocoded result for an area will be (0,0).
CENTER_LAT	NUMBER	Latitude value of the center of the postal-code area. (See the explanation for the CENTER_LONG column.)
ROAD_SEGMENT_ID	NUMBER(10)	ID number of the road segment to which the postal-code area's center is set. This value must be correctly set if the geocoder is intended to work with the Oracle Spatial routing engine; otherwise, it can be set to any non-zero value, but it cannot be null. (Required)
COUNTRY_CODE_2	VARCHAR2(2)	2-letter ISO country code of the country to which the postal-code area belongs. (Required)
PARTITION_ID	NUMBER	Partition key used for partitioning geocoder data by geographic boundaries. If the data is not partitioned, set this value to 1. (Required)
NUM_STREETS	NUMBER	The number of road segments inside the postal-code area.

GC_POSTAL_CODE Index

There is one required index on the GC_POSTAL_CODE_<suffix> table. It is created as illustrated in Example 5.

Example 5: GC_POSTAL_CODE_<suffix> Index

```
CREATE INDEX idx_<suffix>_ postcode ON gc_postal_code_<suffix> (country_code_2,
postal_code);
```

2.1.6 GC_POI Table and Indexes

The GC_POI_<suffix> table (for example, GC_POI_US) stores point-of-interest (POI) information for the country or group of countries associated with the table-name suffix. POIs include features like airports, monuments, parks, etc. This table contains one or more rows for each POI; it may contain multiple rows for a POI when the POI is associated with multiple settlements. The GC_POI_<suffix> table contains the columns shown in Table 6.

Table 6: GC_POI_<suffix> Table

Column Name	Data Type	Description
POI_ID	NUMBER	ID number of the POI. (Required)
POI_NAME	VARCHAR2(64)	Name of the POI. (Required)

Column Name	Data Type	Description
LANG_CODE	VARCHAR2(3)	3-letter ISO national language code for the language of the POI name. (Required)
FEATURE_CODE	NUMBER	Feature code for the POI, if the data vendor classifies POIs by category.
HOUSE_NUMBER	VARCHAR2(10)	House number of the POI; may contain non-numeric characters. (Required)
STREET_NAME	VARCHAR2(80)	Road name of the POI. (Required)
SETTLEMENT_ID	NUMBER(10)	ID number of the settlement to which the POI belongs. (Required if the POI is associated with a settlement)
MUNICIPALITY_ID	NUMBER(10)	ID number of the municipality to which the POI belongs. (Required if the POI is associated with a municipality)
REGION_ID	NUMBER(10)	ID number of the region to which the POI belongs. (Required if the POI is associated with a region)
SETTLEMENT_NAME	VARCHAR2(64)	Name of the settlement to which the POI belongs. (Required if the POI is associated with a settlement)
MUNICIPALITY_NAME	VARCHAR2(64)	Name of the municipality to which the POI belongs. (Required if the POI is associated with a municipality)
REGION_NAME	VARCHAR2(64)	Name of the region to which the POI belongs. (Required if the POI is associated with a region)
POSTAL_CODE	VARCHAR2(16)	Postal code of the POI. (Required)
VANITY_CITY	VARCHAR2(35)	Name of the city popularly associated with the POI, if it is different from the actual city containing the POI. For example, the London Heathrow Airport is actually located in a town named Hayes, which is part of greater London, but people tend to associate the airport only with London. In this case, the VANITY_CITY value is London.
ROAD_SEGMENT_ID	NUMBER	ID number of the road segment on which the POI is located. (Required)
SIDE	VARCHAR2(1)	Side of the road on which the POI is located. Possible values: L (left) or R (right). (Required)
PERCENT	NUMBER	Decimal fraction of the length of the road segment on which the POI is located. It is computed as the distance from the road segment start-point to the POI, divided by the length of the road segment. (Required)
TELEPHONE_NUMBER	VARCHAR2(20)	Telephone number of the POI.

Column Name	Data Type	Description
LOC_LONG	NUMBER	Longitude value of the POI. (Required)
LOC_LAT	NUMBER	Latitude value of the POI. (Required)
COUNTRY_CODE_2	VARCHAR2(2)	2-letter ISO country code for the country to which the POI belongs. (Required)
PARTITION_ID	NUMBER	Partition key used for partitioning geocoder data by geographic boundaries. If the data is not partitioned, set this value to 1. (Required)

GC_POI Indexes

There are four required indexes on the GC_POI_<suffix> table. They are created as illustrated in Example 6.

Example 6: GC_POI_<suffix> Indexes

```
CREATE INDEX idx_<suffix>_poi_name ON gc_poi_<suffix> (country_code_2, name);

CREATE INDEX idx_<suffix>_poi_setnm ON gc_poi_<suffix> (country_code_2,
settlement_id, name);

CREATE INDEX idx_<suffix>_poi_ munnm ON gc_poi_<suffix> (country_code_2,
municipality_id, name);

CREATE INDEX idx_<suffix>_poi_ regnm ON gc_poi_<suffix> (country_code_2,
region_id, name);
```

2.1.7 GC_ADDRESS_POINT Table and Indexes

The GC_ADDRESS_POINT_<suffix> table (for example, GC_ADDRESS_POINT_US) stores the geographic (latitude, longitude) coordinates for addresses in the country or group of countries associated with the table-name suffix. This table is not required for geocoding, however, it enables the Geocoder to provide more accurate location results and is required for point-based geocoding. It is automatically used when present in the schema. The GC_ADDRESS_POINT_<suffix> table contains the columns shown in Table 7.

Table 7: GC_ADDRESS_POINT_<suffix> Table

Column Name	Data Type	Description
ADDRESS_POINT_ID	NUMBER	ID number of the address point.
ROAD_ID	NUMBER	ID number of the road on which the address point is located. (Required)
ROAD_SEGMENT_ID	NUMBER(10)	ID number of the road segment on which the address point is located. (Required)
SIDE	VARCHAR2(1)	Side of the road on which the address point is located. Possible values: L (left) or R (right). (Required)

Column Name	Data Type	Description
LANG_CODE	VARCHAR2(3)	3-letter ISO national language code for the language of the address point. (Required)
HOUSE_NUMBER	VARCHAR2(10)	House number of the address point; may contain non-numeric characters. (Required)
PERCENT	NUMBER	Decimal fraction of the length of the road segment on which the address point is located. It is computed as the distance from the road segment start-point to the address point, divided by the length of the road segment. (Required)
ADDR_LONG	NUMBER	Longitude value of the address point. (Required)
ADDR_LAT	NUMBER	Latitude value of the address point. (Required)
COUNTRY_CODE_2	VARCHAR2(2)	2-letter ISO country code for the country to which the address point belongs. (Required)
PARTITION_ID	NUMBER	Partition key used for partitioning geocoder data by geographic boundaries. If the data is not partitioned, set this value to 1. (Required)

GC_ADDRESS_POINT Indexes

There is one required indexes on the GC_ADDRESS_POINT_<suffix> table. It is created as illustrated in Example 7.

Example 7: GC_ADDRESS_POINT_<suffix> Indexes

```
CREATE INDEX idx_<suffix>_addrpt_addr ON gc_address_point_<suffix>
(road_segment_id, road_id, house_number, side);
```

2.2 The Profile Tables

The Oracle Geocoder uses a profile-driven approach to flexibly geocode international addresses without altering the data tables for different countries. The profile tables, GC_COUNTRY_PROFILE, GC_PARSER_PROFILES, and GC_PARSER_PROFILE-AFS are pivotal to interpreting addresses entered for geocoding, since they store address-format definitions for all supported countries. The structure and data required for the three profile tables are described in this section:

2.2.1 GC_COUNTRY_PROFILE Table

The GC_COUNTRY_PROFILE table stores country-profile information used by the Oracle Geocoder. This includes administrative-area hierarchy definitions, the national languages and the table-name suffix used by the seven data tables and their indexes. The GC_COUNTRY_PROFILE table stores one row for each supported country and contains the columns shown in Table 8.

Table 8: GC_COUNTRY_PROFILE Table

Column Name	Data Type	Description
COUNTRY_NAME	VARCHAR2(60)	Complete name of the country. (Required)
COUNTRY_CODE_3	VARCHAR2(3)	3-letter ISO country code for the country. (Required)
COUNTRY_CODE_2	VARCHAR2(2)	2-letter ISO country code for the country. (Required)
LANG_CODE_1	VARCHAR2(3)	3-letter ISO national language code. Some countries may have multiple national languages, in which case LANG_CODE_2 and other LANG_CODE columns should contain values as appropriate. (Required)
LANG_CODE_2	VARCHAR2(3)	3-letter ISO national language code for the country.
LANG_CODE_3	VARCHAR2(3)	3-letter ISO national language code for the country.
LANG_CODE_4	VARCHAR2(3)	3-letter ISO national language code for the country.
NUMBER_ADMIN_LEVELS	NUMBER(1)	Number of administrative levels for the country. A country can have up to 7 administrative-area levels, numbered from 1 to 7: the largest administrative area to the smallest administrative area, respectively. The top-level area (country) is level 1. For the US, the administrative hierarchy is as follows: level 1 = country, level 2 = state, level 3 = county, level 4 = city. (Required)
SETTLEMENT_LEVEL	NUMBER(1)	Administrative level for the settlement, which is the lowest area level or smallest area used in addressing. In the US, this is the city, level 4. In Europe, this is generally a subdivision of a city, level 5 (Required)
MUNICIPALITY_LEVEL	NUMBER(1)	Administrative level for the municipality, which is the second-lowest area level or second-smallest area used in addressing. In the US, this is the county, level 3. In Europe, this is generally a city, level 4.
REGION_LEVEL	NUMBER(1)	Administrative level for the region, which is above the municipality level. In the US, this is the state or third-lowest area level used in addressing, level 2. In Europe, this is a recognized subdivision of the country, level 2 or level 3.

Column Name	Data Type	Description
SETTLEMENT_IS_ OPTIONAL	VARCHAR2(1)	Contains F if settlement information is required in the address data and T otherwise. (Required)
MUNICIPALITY_IS_ OPTIONAL	VARCHAR2(1)	Contains F if municipality information is required in the address data and T otherwise. (Required)
REGION_IS_OPTIONAL	VARCHAR2(1)	Contains F if region information is required in the address data and T otherwise. (Required)
POSTCODE_IN_ SETTLEMENT	VARCHAR(1)	Contains T if each postal code must be completely within a settlement area; contains F if a postal code can include areas from multiple settlements. (Required)
SETTLEMENT_AS_CITY	VARCHAR(1)	Contains T if a city name can identify both a municipality and a settlement; contains F if a city name can only identify a settlement. For example, in the United Kingdom, London can be both the name of a municipality area and the name of a settlement area, which is inside the municipality of London. This is common in large cities in European countries like the UK and Belgium. (Required)
CACHED_ADMIN_AREA_ LEVEL	NUMBER	(Reserved for future use.)
GC_TABLE_SUFFIX	VARCHAR2(5)	Table-name suffix used on the 7 GC_* data tables. For example, if the value of GC_TABLE_SUFFIX is 'US', the names of the geocoding data tables must end with _US (for example, GC_ROAD_SEGMENT_US, CG_AREA_US, etc.). (Required)
CENTER_LONG	NUMBER	Longitude value of the center of the area represented by the geocoding data.
CENTER_LAT	NUMBER	Latitude value of the center of the area represented by the geocoding data.
SEPARATE_PREFIX	VARCHAR2(1)	Contains T if the street-name prefix is a separate word from the street name; contains F if the street name prefix is in the same word with the street name. For example, in a US street address of 123 N Main St, the prefix is N, and it is separate from the street name, which is Main. (Not currently used by Oracle)

Column Name	Data Type	Description
SEPARATE_SUFFIX	VARCHAR2(1)	Contains T if the street-name suffix is a separate word from the street name; contains F if the street-name suffix is in the same word with the street name. For example, in a US street address of 123 Main St NW, the suffix is NW, and it is separate from the street name, which is Main, and from the street type, which is St. (Not currently used by Oracle)
SEPARATE_STYPE	VARCHAR2(1)	Contains T if the street type is a separate word from the street name; contains F if the street type is in the same word with the street name. For example, in a German street address of 123 Beethovenstrasse, the type is strasse, and it is in the same word with the street name, which is Beethoven. (Not currently used by Oracle)
AREA_ID	NUMBER	Not currently used by Oracle.
VERSION	VARCHAR2(10)	Version of the data. The current version is 1.0. (Required)

2.2.2 GC_PARSER_PROFILES Table

The Oracle Geocoder uses keywords to identify address fields, such as, house number, road name, city name, state name, and zip code. The GC_PARSER_PROFILES table stores information about keywords typically found in postal addresses. A keyword can be the type of street, for example, road, street, drive, avenue, etc. or the prefix or suffix of a street, for example, north, south, east, west, etc. The GC_PARSER_PROFILES table contains the columns shown in Table 9.

Table 9: GC_PARSER_PROFILES Table

Column Name	Data Type	Description
COUNTRY_CODE	VARCHAR2(2)	2-letter ISO country code for the keyword. (Required)

Column Name	Data Type	Description
KEYWORDS	SDO_KEYWORDARRAY	A single array of keywords that will be identified by the Oracle Geocoder for a specific address field. The array may contain a single word, or a group of words and abbreviations that can be used with the same meaning, for example, United States of America, USA, and United States all refer to the US. The first word of this array should be the official full name of the keyword, if there is any. The US uses over 400 keywords in parsing addresses. Here are some examples of keyword arrays and keywords from the US dataset. However, only a single SDO_KEYWORDARRAY is stored per row: SDO_KEYWORDARRAY('UNITED STATES OF AMERICA','US', 'USA', 'UNITED STATES', 'U.S.A.', 'U.S.') SDO_KEYWORDARRAY('AVENUE','AV', 'AVE', 'AVEN', 'AVENU', 'AVN', 'AVNUE', 'AV.','AVE.') SDO_KEYWORDARRAY('40TH', 'FORTIETH') SDO_KEYWORDARRAY('NEW YORK','NY') SDO_KEYWORDARRAY('LIBRARY')
OUTPUT_KEYWORD	VARCHAR2(2000)	A keyword used in the geocoder data to represent an address field. It must be the same as one of the keywords used in the keyword array. The output keyword is used to match the addresses stored in the geocoding data tables to the users' input, for example, if the output_keyword 'AV' is used for street type Avenue in the GC_ROAD_US table, wherever a user enters an address containing any of the keywords ('AVENUE','AV', 'AVE', 'AVEN', 'AVENU', 'AVN', 'AVNUE', 'AV.','AVE.'), the keyword will be interpreted and matched to the output keyword AV to help find the address in the database. Here are the output keywords for the above keyword arrays. However, only a single OUTPUT_KEYWORD is stored per row: US AV 40TH NY LIBRARY

Column Name	Data Type	Description
SECTION_LABEL	VARCHAR2(30)	A label used to identify the type of keyword represented in the KEYWORDS and OUTPUT_KEYWORD columns. There are the nine different labels. However, only a single SECTION_LABEL per row is used in identifying the type of keywords: COUNTRY_NAME: Identifies keywords that are used to represent country names. LOCALITY_KEYWORD_DICTIONARY: Identifies keywords that are used to replace words in a locality (city, state, province, etc.) with a standardized form of the word, for example, Saint replaced by St; by doing so, the city names "Saint Thomas" and "St. Thomas" will be standardized to "St Thomas", which is stored in the database. PLACE_NAME_KEYWORD: Identifies a POI name keyword, for example, a restaurant, hotel, etc. REGION_LIST: Identifies keywords that are known names of regions, for example, NY, New York, NH, New Hampshire, etc. The regions identified must be administrative areas that belong to the third-lowest area level or third-smallest area used in addressing. In the US, this is the state level – the lowest area level or smallest area is the city level. SECOND_UNIT_KEYWORD: Identifies keywords used in second-unit descriptions, such as Floor, #, Suite, Apartment, etc. STREET_KEYWORD_DICTIONARY: Identifies keywords used to replace non-street-type keywords in street names, with a standardized form, for example: 40TH, Fortieth STREET_PREFIX_KEYWORD: Identifies street name prefix keywords such as South, North, West, East. STREET_TYPE_KEYWORD: Identifies street type keywords such as Road, Street, Drive, etc. IN_LINE_STREET_TYPE_KEYWORD: Identifies street type keywords that are attached to street names, for example, strasse in the German street name Steinstrasse.
POSITION	VARCHAR2(1)	The position of the keyword relative to a street name. It tells the Geocoder whether the keyword can precede (P) or follow (F) the actual street name, or both (B). P, F, and B are therefore the only valid entries. In the US, most street type keywords follow the street names, for example, the street type Blvd in Hollywood Blvd. In France, however, street type keywords usually precede the street names, for example, the street type Avenue in Avenue De Paris.

Column Name	Data Type	Description
SEPARATENESS	VARCHAR2(1)	An indicator for whether or not the keyword is separate from a street name. Keywords are either separable (S) or non-separable (N). S and N are therefore the only valid entries. In the US, all street-type keywords are separate words from the street name, for example, the street type Blvd in Hollywood Blvd. In Germany, however, the street-type keywords are not separate from the street name, for example, the street type strasse in Augustenstrasse.

2.2.3 GC_PARSER_PROFILEAFS Table

The GC_PARSER_PROFILEAFS table stores the XML definition of postal-address formats. An XML string describes each address format for a specific country. In the Oracle Geocoder 10g and earlier, the J2EE geocoder uses a `country_name.ppr` file instead of this table. The content of the `country_name.ppr` file is equivalent to the content of the ADDRESS_FORMAT_STRING attribute. The GC_PARSER_PROFILESAFS table contains the columns shown in Table 10, and the XML definition of the ADDRESS_FORMAT_STRING is shown in Example 8.

Table 10: GC_PARSER_PROFILEAFS Table

Column Name	Data Type	Description
COUNTRY_CODE	VARCHAR2(2)	2- letter ISO country code. (Required)
ADDRESS_FORMAT_STRING	CLOB	XML string describing the address format for the country specified in the COUNTRY_CODE column. (See Example 8.)

Example 8: XML Definition for the US Address Format

```
<address_format unit_separator="," replace_hyphen="true">
 <address_line>
   <place_name />
 </address_line>
 <address_line>
   <street_address>
     <house_number>
       <format form="0*" effective="0-1" output="$" />
       <format form="0*1*" effective="0-1" output="$">
         <exception form="0*TH" />
         <exception form="0*ST" />
         <exception form="0*ND" />
         <exception form="0*RD" />
       </format>
       <format form="0*10*" effective="0-1" output="$" />
       <format form="0*-0*" effective="0-1" output="$" />
       <format form="0*.0*" effective="0-1" output="$" />
       <format form="0* 0*/0*" effective="0-1" output="$" />
     </house_number>
     <street_name>
       <prefix />
       <base_name />
      <suffix />
```

```
      <street_type />
      <special_format>
         <format form="1* HWY 0*" effective="7-8" addon_effective="0-1"
      addon_output="$ HWY"/>
         <format form="1* HIGHWAY 0*" effective="11-12" addon_effective="0-1"
      addon_output="$ HWY"/>
         <format form="1* HWY-0*" effective="7-8" addon_effective="0-1"
      addon_output="$ HWY"/>
         <format form="1* HIGHWAY-0*" effective="11-12" addon_effective="0-1"
      addon_output="$ HWY"/>
        <format form="HWY 0*" effective="4-5" addon_output="HWY" />
        <format form="HIGHWAY 0*" effective="8-9" addon_output="HWY" />
        <format form="ROUTE 0*" effective="6-7" addon_output="RT" />
        <format form="I 0*" effective="2-3" addon_output="I" />
        <format form="11 0*" effective="3-4" addon_effective="0-1" />
        <format form="I0*" effective="1-2" addon_output="I" />
        <format form="I-0*" effective="2-3" addon_output="I" />
        <format form="11-0*" effective="3-4" addon_effective="0-1" />
        <format form="ROUTE-0*" effective="6-7" addon_output="RT" />
        <format form="US0*" effective="2-3" addon_output="US" />
        <format form="HWY-0*" effective="2-3" addon_output="US" />
        <format form="HIGHWAY-0*" effective="8-9" addon_output="HWY" />
       <format form="$[PF] 0* $[SF]" effective="6-13" output="$" />
      </special_format>
    </street_name>
    <second_unit>
      <special_format>
        <format form="# 0*" effective="2-3" output="APT $" />
        <format form="#0*" effective="1-2" output="APT $" />
      </special_format>
    </second_unit>
  </street_address>
 </address_line>
 <address_line>
  <po_box>
    <format form="PO BOX 0*" effective="7-8" />
    <format form="P.O. BOX 0*" effective="9-10" />
    <format form="PO 0*" effective="3-4" />
    <format form="P.O. 0*" effective="5-6" />
    <format form="POBOX 0*" effective="6-7" />
  </po_box>
 </address_line>
 <address_line>
  <city optional="no" />
  <region optional="no" order="1" />
  <postal_code>
    <format form="00000" effective="0-4" />
    <format form="00000-0000" effective="0-4" addon_effective="6-9" />
    <format form="00000 0000" effective="0-4" addon_effective="6-9" />
  </postal_code>
 </address_line>
</address_format>
```

ADDRESS_FORMAT_STRING

The XML address-format string describes the format of address fields and their positioning in valid postal addresses. The address-format string is organized by address lines, since postal addresses are typically written in multiple address lines.

<address_format> element

The `<address_format>` element includes the `unit_separator` and `replace_hyphen` attributes. The `unit_separator` is used to separate fields in the stored data. By default it is a comma, i.e. `unit_separator=",”`. The `replace_hyphen` attribute specifies whether to replace all hyphens in the users' input with a space. By default it is set to `true`, i.e. `replace_hyphen="true”`. When `true`, it is expected that all names in the data tables will contain a space instead of a hyphen. Administrative-area names in the data tables containing hyphens will not be matched during geocoding if `replace_hyphen="true”`. However, these area names with hyphens can be placed in the REAL_NAME column of the GC_AREA table to be returned as the administrative-area name in the geocoded result. Road names in the NAME column of the GC_ROAD table containing hyphens will, however, be matched during geocoding if `replace_hyphen="true”`, but the matching performance will be degraded.

<address_line> elements

Each `<address_line>` element in the XML address format string describes the format of an address line. Each `<address_line>` can have one or more child elements describing the individual address fields, such as, street address, city, state (region or province) and postal code. These address field elements are listed in the same order as the address fields appear in valid postal addresses. The `optional` attribute of the address field element is set to `“no”` if the address field is mandatory. By default, address field elements are optional.

<format> elements

The format descriptions for house number, special street name, post box and postal code elements are specified with a single or multiple `<format>` elements. Each `<format>` element specifies a valid layout and range of values for a particular address field. The following example illustrates the format used to define a special street name:

```
<format
     form="1* HWY 0*"
     effective="7-8"
     output="$"
     addon_effective="0-1"
     addon_output="$ HIGHWAY" />
```

The `form` attribute uses a regular expression-like string to describe the format: `1` stands for any alphabetic letter; `0` stands for any numerical digit; `2` stands for any alphabetic letter or any numerical digit; `1*` specifies a sting consisting of all alphabetic letters; `0*` specifies a sting consisting of all numerical digits; `2*` specifies a string consisting of numerical digits and/or alphabetic letters. All other symbols represent themselves.

Any string matching the pattern specified by the `form` attribute is considered by the Oracle Geocoder to be a valid string for its (parent) address field. A valid string can then be broken down into segments specified by the attributes, `effective` and `addon_effective`. The `effective` attribute specifies a sub-string of the full pattern using the start and end positions for the end descriptor of the `form` attribute. In

the above example, effective="7-8" retrieves the sub-string (counting from position 0) starting at position 7 and ending at position 8, which is the sub-string defined by 0*, at the end of the form attribute. The addon_effective attribute specifies a sub-string of the full pattern using the start and end positions for the start descriptor of the form attribute. In the above example, addon_effective="0-1" retrieves the sub-string, (counting from position 0) starting at position 0 and ending at position 1, which is the sub-string defined by 1*, at the beginning of the form attribute. The effective attribute specifies the more important, primary piece of the address string; the addon_effective attribute specifies the secondary piece of the address string.

The output and addon_output attributes specify the output form of the address string for segments specified by effective and addon_effective, respectively. The Oracle Geocoder uses these output forms during address matching. The symbol $ stands for the matched string and other symbols represent themselves. In the above example, output="$", the $ stands for the sub-string that was matched in the effective attribute; addon_output="$ HIGHWAY" stands for the sub-string that was matched in the addon_effective attribute, followed by a space, followed by the word "HIGHWAY".

Using the <format> element in the above example, with form="1* HWY 0*", the input string 'STATE HWY 580' will have: effective=580; output=580; addon_effective=STATE; and addon_output=STATE HIGHWAY.

The <format> element may also contain an <exception> sub-element. The <exception> sub-element specifies a string that has a valid form, but must be excluded from the address field. In a <house_number> element with valid numbers 0*1*, that is, any numeric digits followed by any alphabetic letters, <exception form="0*TH" /> specifies that any house number with (or without) numeric digits and ending with "TH" must be excluded.

The Oracle Geocoder address parser uses the format description defined in the XML address format, combined with the keyword definition for each address field defined in the GC_PARSER_PROFILES table to parse the input address and identify individual address fields.

2.2.4 Installing the Profile Tables

The Oracle Geocoder profile tables are typically supplied by a data provider. Use the data provider's profile tables for geocoding whenever they are available. For users building their own geocoder schema, Oracle provides sample GC_COUNTRY_PROFILE, GC_PARSER_PROFILES and GC_PARSER_PROFILEAFS tables. The installation of these Oracle-supplied profile tables should only be undertaken if profile tables are not supplied with the data tables.

The Oracle-supplied tables contain parser profiles for a limited number of countries. If profiles for your country or group of countries of interest are not included, you will need to manually add them using the information provided in Sections 2.2.1, 2.2.2 and 2.2.3. For a quick start, you may copy the parser profiles of a country with a similar address format to your country of interest, and edit these profiles where necessary. If your parser

profiles of interest are included in the Oracle-supplied tables, you may use them directly or update them if necessary. No sample country profiles are provided, so you will need to add your own.

To install and query the Oracle-supplied profile tables, perform the following steps:

1. Log on to your database as the `geocoder user`. The geocoder user is the user under whose schema the geocoder schema will be loaded.
2. Create the GC_COUNTRY_PROFILE, GC_PARSER_PROFILES and GC_PARSER_PROFILEAFS tables by calling the SDO_GCDR.CREATE_PROFILE_TABLES() procedure:

   ```
   > CALL SDO_GCDR.CREATE_PROFILE_TABLES();
   ```

3. Populate the GC_PARSER_PROFILES and GC_PARSER_PROFILEAFS tables by running the `sdogcprs.sql` script found in your $ORACLE_HOME/md/admin subdirectory:

   ```
   > @$ORACLE_HOME/md/admin/sdogcprs.sql
   ```

4. Query the parser profile tables to determine if parser profiles for your country of interest are supplied:

   ```
   > SELECT DISTINCT(country_code) FROM gc_parser_profiles;
   > SELECT DISTINCT(country_code) FROM gc_parser_profileafs;
   ```

3 Using the Oracle Geocoder

The Oracle Geocoder is supported in three forms: the database geocoder, the J2EE geocoder and the Thin-Client geocoder. The database geocoder uses PL/SQL APIs to access the geocoder schema; the J2EE geocoder uses an XML API to provide a service that accesses the geocoder schema; and the Thin-Client geocoder uses client-side Java APIs to access the geocoder schema. The geocoder schema is typically supplied by a data provider and is generally ready to use out-of-the-box. Refer to the data provider's documentation for installation of the geocoder schema, which must be installed before the geocoders can be used. The choice of which geocoder to use depends on your application. Web-based applications favor the J2EE geocoder.

3.1 Using the Database Geocoder

The database geocoder is ready for use once the geocoder schema has been loaded into the database. Addresses are then geocoded by calling subprograms of the SDO_GCDR PL/SQL package. The addresses are entered into these subprograms using specific geocoding datatypes and the geocoded results are also returned in these datatypes. There are three datatypes that must be understood in order to enter addresses and extract and interpret geocoding results.

The SDO_GEO_ADDR type, the SDO_KEYWORDARRAY type, and the SDO_ADDR_ARRAY type are the datatypes used in database geocoding. Addresses to be geocoded are represented either as formatted addresses or unformatted addresses. The SDO_GEO_ADDR type is used to describe formatted addresses, while the SDO_KEYWORDARRAY is used to describe unformatted addresses. The

SDO_ADDR_ARRAY type stores multiple SDO_GEO_ADDR objects and is used when multiple addresses are returned. A description of these datatypes and the SDO_GCDR PL/SQL package follows:

3.1.1 SDO_KEYWORDARRAY Type

The SDO_KEYWORDARRAY datatype is used to store the address lines of an unformatted address. An unformatted address is specified using strings with address information in the postal-address format of the country to which the address belongs. For example, an unformatted US address may consist of the following strings: '22 Monument Square' and 'Concord, MA 01742'. For database geocoding, these strings are stored in the SDO_KEYWORDARRAY type. The SDO_KEYWORDARRAY type is a VARRAY (variable length array) of VARCHAR2 strings and is defined as follows:

```
CREATE TYPE sdo_keywordarray AS VARRAY(10000) OF VARCHAR2(9000);
```

3.1.2 SDO_GEO_ADDR Type

The SDO_GEO_ADDR datatype is used to describe a formatted address. A formatted address is defined by a specific set of attributes, which may include the street name, settlement, postal code, and country. The SDO_GEO_ADDR attributes used to input an address or that are returned in a geocoded result depend on factors related to the address, in particular, the country to which it belongs. Table 11 lists the attributes of the SDO_GEO_ADDR type, though not all of these attributes will be used in any given case. Detailed descriptions of MATCHMODE, MATCHCODE, ERRORMESSAGE and MATCHVECTOR attributes follow Table 11.

Table 11: SDO_GEO_ADDR Type Attributes for Formatted Address Input and Output

Attribute	**Data Type**	**Description**
ID	NUMBER	(Not currently used.)
ADDRESSLINES	SDO_KEYWORDARRAY	Address strings stored in the SDO_KEYWORDARRAY type.
PLACENAME	VARCHAR2(200)	Point-of-interest (POI) name. Example: California Pacific Medical Ctr
STREETNAME	VARCHAR2(200)	Street name, including street type. Example: Main St
INTERSECTSTREET	VARCHAR2(200)	Intersecting street.
SECUNIT	VARCHAR2(200)	Secondary unit, such as an apartment number or building number.

Attribute	Data Type	Description
SETTLEMENT	VARCHAR2(200)	The lowest-level or smallest administrative area to which the address belongs. In most cases it is the city. In some European countries, the settlement can be an area within a large city, in which case the large city is the municipality.
MUNICIPALITY	VARCHAR2(200)	The administrative area above a settlement. Municipality is not used for US addresses. In European countries where cities contain settlements, the municipality is the city.
REGION	VARCHAR2(200)	The administrative area above a municipality (if applicable), or above a settlement if a municipality does not apply. In the US, the region is the state; in some other countries, the region is the province.
COUNTRY	VARCHAR2(100)	Country name or ISO country code.
POSTALCODE	VARCHAR2(20)	Postal code (optional if the administrative area information is provided). In the US, the postal code is the 5-digit ZIP code.
POSTALADDONCODE	VARCHAR2(20)	String appended to the postal code. In the US, the postal add-on code is typically the last four numbers of a 9-digit ZIP code specified in a 5-4 format.
FULLPOSTALCODE	VARCHAR2(20)	Full postal code, including the postal code and postal add-on code.
POBOX	VARCHAR2(100)	Post-office box number.
HOUSENUMBER	VARCHAR2(100)	House or building number. Example: 123 in 123 Main St
BASENAME	VARCHAR2(200)	Base name of the street. Example: Main in 123 Main St
STREETTYPE	VARCHAR2(20)	Type of the street. Example: St in 123 Main St
STREETTYPEBEFORE	VARCHAR2(1)	(Not currently used.)
STREETTYPEATTACHED	VARCHAR2(1)	(Not currently used.)
STREETPREFIX	VARCHAR2(20)	Prefix for the street. Example: S in 123 S Main St

Attribute	Data Type	Description
STREETSUFFIX	VARCHAR2(20)	Suffix for the street. Example: NE in 123 Main St NE
SIDE	VARCHAR2(1)	Side of the road (L for left or R for right) that the house is on when you are traveling along the road segment following its orientation (that is, from its start node toward its end node). The house numbers may be increasing or decreasing.
PERCENT	NUMBER	Decimal fraction indicating how far along the road segment the address is, when traveling in the direction of the orientation of the road segment.
EDGEID	NUMBER	Edge ID of the road segment.
ERRORMESSAGE	VARCHAR2(20)	Error message (see below). Note: The MATCHVECTOR attribute supersedes this attribute.
MATCHCODE	NUMBER	Match code indicates which data was matched (see below).
MATCHMODE	VARCHAR2(30)	Match mode determines how closely an address must match the data used for geocoding (see below).
LONGITUDE	NUMBER	Longitude coordinate value.
LATITUDE	NUMBER	Latitude coordinate value.
MATCHVECTOR	VARCHAR2(20)	A string that indicates how each address attribute has been matched against the data used for geocoding (see below).

ERRORMESSAGE Attribute

The error message attribute contains a string that indicates which input address attributes matched the data stored in the geocoder schema. The value of the string is set to '???????????281C??' before the geocoding operation begins. It is then modified to reflect which address attributes were matched during the geocoding operation. Table 11.1 lists the positions in the string and the address attribute corresponding to each position. It also lists the character value to which the position in the string is set, if the address attribute is matched. The ERRORMESSAGE attribute has been superseded by the MATCHVECTOR attribute, but retained for backward compatibility.

Table 11.1: Geocoded Address Error Message Interpretation

Position	Attribute	Value if Matched
1-2	Reserved for future use	??
3	Address point	X
4	POI name	O
5	House or building number	#
6	Street prefix	E
7	Street base name	N
8	Street suffix	U
9	Street type	T
10	Secondary unit	S
11	Built-up area or city	B
12-13	(Reserved)	(Ignore any values in these positions.)
14	Region	1
15	Country	C
16	Postal code	P
17	Postal add-on code	A

MATCHCODE Attribute

The match code is a number indicating which input address attributes matched the data stored in the geocoder schema. The match code is returned in an output SDO_GEO_ADDR object. Table 11.2 lists the possible match code values.

Table 11.2: MATCHCODE Values for Geocoding Operations

MATCHCODE Value	Description
1	Exact match: the city name, postal code, street base name, street type (and suffix or prefix or both, if applicable), and house or building number match the data in the geocoder schema.
2	The city name, postal code, street base name, and house or building number match the data in the geocoder schema, but the street type suffix or prefix does not match.

MATCHCODE Value	Description
3	The city name, postal code, and street base name match the data in the geocoder schema, but the house or building number does not match.
4	The city name and postal code match the data in the geocoder schema, but the street address does not match.
10	The city name matches the data in the geocoder schema, but the postal code does not match.
11	The postal code matches the data in the geocoder schema, but the city name does not match.

MATCHMODE Attribute

The match mode for a geocoding operation determines how closely the attributes of an input address must match the data stored in the geocoder schema. Input addresses may use different representations for a specific part of an address (such as Street and the abbreviation St); and they may include minor errors (such as the wrong postal code, even though the street name and city are correct). You may require an exact match between the input address and the data used for geocoding, or you may relax the requirements for some attributes so that geocoding can be performed despite discrepancies in the input address.

Table 11.3 lists the match modes and their meanings. As the match mode value moves from RELAX_STREET_TYPE to RELAX_ALL, each mode includes all of the characteristics of the previously listed mode. For example, RELAX_POI_NAME includes all the characteristics of RELAX_STREET_TYPE, and RELAX_HOUSE_NUMBER includes all the characteristics of RELAX_POI_NAME, etc. Use a value from Table 11.3 in the MATCHMODE attribute of the SDO_GEO_ADDR data type (described above) and in the `match_mode` parameter of a geocoding function or procedure.

Table 11.3: MATCHMODE Values for Geocoding Operations

MATCHMODE Value	Description
EXACT	All attributes of the input address must match the data stored in the geocoder schema. However, if the house or building number, base name (street name), street type, street prefix, and street suffix do not all match the geocoding data, a location in the first match found in the following is returned: postal code, city or town (settlement) within the state, and state. For example, if the street name is incorrect but a valid postal code is specified, a location in the postal code is returned.
RELAX_STREET_TYPE	The street type can be different from the data used for geocoding. For example, if Main St is stored in the geocoder schema, Main Street would also match that, as would Main Blvd, even if there was no Main Blvd and no other street type for Main in the data used for geocoding.

MATCHMODE Value	Description
RELAX_POI_NAME	The name of the POI does not have to match the data used for geocoding. For example, if Jones State Park is in the geocoder schema, Jones State Pk and Jones Park would also match, as long as there were no ambiguities or other matches in the data.
RELAX_HOUSE_NUMBER	The house or building number and street type can be different from the data used for geocoding. For example, if 123 Main St is in the geocoder schema, 123 Main Lane and 124 Main St would also match, as long as there were no ambiguities or other matches in the data.
RELAX_BASE_NAME	The base name of the street, the house or building number, and the street type can be different from the data used for geocoding. For example, if Pleasant Valley is the base name of a street in the geocoder schema, Pleasant Vale would also match, as long as there were no ambiguities or other matches in the data.
RELAX_POSTAL_CODE	The postal code (if provided), base name, house or building number, and street type can be different from the data used for geocoding.
RELAX_BUILTUP_AREA	The address can be outside the city specified, as long as it is within the same county. Also includes the characteristics of RELAX_POSTAL_CODE.
RELAX_ALL	Equivalent to RELAX_BUILTUP_AREA.
DEFAULT	Equivalent to RELAX_POSTAL_CODE.

MATCHVECTOR Attribute

The match vector attribute contains a string that indicates which input address attributes matched the data stored in the geocoder schema. It provides a more accurate and detailed description of the match for each attribute, than the ERRORMESSAGE attribute. It is intended to supersede the error message attribute, which has been kept for backward compatibility. The value of the match vector string is set to '??????????????????' before the geocoding operation begins. It is then modified to reflect which address attributes were matched during the geocoding operation. Table 11.4 lists the positions in the string and the address attribute corresponding to each position. The numeric value to which the position in the string is set may vary from 0-4. A description of these values is given in Table 11.4.1.

Table 11.4: Geocoded Address Match Vector Interpretation

Position	Attribute
1-2	Reserved for future use
3	Address point
4	POI name

Position	Attribute
5	House or building number
6	Street prefix
7	Street base name
8	Street suffix
9	Street type
10	Secondary unit
11	Built-up area or city
12-13	(Reserved for future use)
14	Region
15	Country
16	Postal code
17	Postal add-on code

Table 11.4.1: MATCHVECTOR Values for Geocoded Address

Value	Description
0	The input attribute is not null and is matched with a non-null value.
1	The input attribute is null and is matched with a null value.
2	The input attribute is not null and is replaced by a different non-null value.
3	The input attribute is not null and is replaced by a null value.
4	The input attribute is null and is replaced by a non-null value.

3.1.3 Accessing Attributes of the SDO_GEO_ADDR Type

The subprograms of the SDO_GCDR PL/SQL geocoding package can return the entire SDO_GEO_ADDR object type, or they can return specific attributes. Example 9, shows statements that geocode the address of the San Francisco City Hall at 1 Carlton B Goodlett Pl, San Francisco, CA 94102. The first two statements return the entire SDO_GEO_ADDR object, and the remaining statements return the specified attributes of the object. The SDO_GEO_ADDR object is the preferred structure for inputting addresses to be geocoded, since it explicitly states where each part of the address belongs.

Example 9: Database Geocoder Returning the SDO_GEO_ADDR Type and Specified Attributes of the Type

```
SQL> SELECT SDO_GCDR.GEOCODE_ADDR('GEOCODER_US',
 2  SDO_GEO_ADDR(NULL, SDO_KEYWORDARRAY(), null, 'Carlton B Goodlett Pl',
 3  NULL, NULL, 'San Francisco', NULL, 'CA', 'US', '94102', NULL, NULL, NULL,
 4  '1', NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
 5  'RELAX_BASE_NAME', NULL, NULL, NULL)) FROM DUAL;

SDO_GCDR.GEOCODE_ADDR('GEOCODER_US',SDO_GEO_ADDR(NULL,SDO_KEYWORDARRAY(),NULL,'
C
--------------------------------------------------------------------------------
SDO_GEO_ADDR(0, SDO_KEYWORDARRAY(), NULL, 'CARLTON B GOODLETT PL', NULL, NULL,
'SAN FRANCISCO', NULL, 'CA', 'US', '94102', NULL, '94102', NULL, '1', 'CARLTON
B GOODLETT', 'PL', 'F', 'F', NULL, NULL, 'L', .01, 23614360,
'????#ENUT?B281CP?', 1, 'RELAX_BASE_NAME', -122.41815, 37.7784183,
'????0101010??000?')

SQL> SELECT SDO_GCDR.GEOCODE('GEOCODER_US',
  2  SDO_KEYWORDARRAY('1 Carlton B Goodlett Pl', 'San Francisco, CA  94102'),
  3  'US', 'RELAX_BASE_NAME')
  4  FROM DUAL;

SDO_GCDR.GEOCODE('GEOCODER_US',SDO_KEYWORDARRAY('1CARLTONBGOODLETTPL','SANFRANC
ISCO
--------------------------------------------------------------------------------
SDO_GEO_ADDR(0, SDO_KEYWORDARRAY(), NULL, 'CARLTON B GOODLETT PL', NULL, NULL,
'SAN FRANCISCO', NULL, 'CA', 'US', '94102', NULL, '94102', NULL, '1', 'CARLTON
B GOODLETT', 'PL', 'F', 'F', NULL, NULL, 'L', .01, 23614360,
'????#ENUT?B281CP?', 1, 'RELAX_BASE_NAME', -122.41815, 37.7784183,
'????0101010??000?')

SQL> SELECT SDO_GCDR.GEOCODE('GEOCODER_US',
  2  SDO_KEYWORDARRAY('1 Carlton B Goodlett Pl', 'San Francisco, CA  94102'),
  3  'US', 'RELAX_BASE_NAME').EDGEID
  4  FROM DUAL;

SDO_GCDR.GEOCODE('GEOCODER_US',SDO_KEYWORDARRAY('1CARLTONBGOODLETTPL','SANFRANC
ISCO
--------------------------------------------------------------------------------
                                                                      23614360

SQL> SELECT G.GC.STREETTYPE, G.GC.SIDE, G.GC.PERCENT, G.GC.EDGEID,
G.GC.MATCHCODE, G.GC.MATCHVECTOR
  2  FROM (
  3  SELECT SDO_GCDR.GEOCODE('GEOCODER_US',
  4  SDO_KEYWORDARRAY('1 Carlton B Goodlett Pl', 'San Francisco, CA 94102'),
  5  'US','RELAX_BASE_NAME') GC
  6  FROM DUAL) G;

GC.STREETTYPE        G GC.PERCENT  GC.EDGEID GC.MATCHCODE GC.MATCHVECTOR
-------------------- - ---------- ---------- ------------ --------------------
PL                   L          0   23614360            1 ????0101010??000?
```

```
SQL> SELECT ID, MATCHVECTOR, LONGITUDE, LATITUDE FROM TABLE (
  2  SDO_GCDR.GEOCODE_ALL('GEOCODER_US',
  3  SDO_KEYWORDARRAY('1 Carlton B Goodlett Pl', 'San Francisco, CA 94102'),
  4  'US','RELAX_BASE_NAME')
  5  );

        ID MATCHVECTOR           LONGITUDE   LATITUDE
---------- -------------------- ---------- ----------
         1 ????0101010??000?    -122.41815    37.7784
```

3.1.4 SDO_ADDR_ARRAY Type

The SDO_ADDR_ARRAY type is a VARRAY (variable length array) of SDO_GEO_ADDR objects used to store geocoded address results. Multiple address objects can be returned when multiple addresses are matched as a result of a geocoding operation. The SDO_ADDR_ARRAY type is defined as follows:

```
CREATE TYPE sdo_addr_array AS VARRAY(1000) OF sdo_geo_addr;
```

3.1.5 SDO_GCDR: Geocoding Package

The MDSYS.SDO_GCDR package contains subprograms for database geocoding address data. The geocoding subprograms are listed in Table 12. Use the SQL command: DESCRIBE SDO_GCDR to see the signature for these subprograms.

Table 12: SDO_GCDR Subprograms

Subprogram	**Description**
SDO_GCDR.GEOCODE	Geocodes an unformatted address and returns an SDO_GEOR_ADDR object.
SDO_GCDR.GEOCODE_ADDR	Geocodes an input address using attributes in an SDO_GEO_ADDR object, and returns the first matched address as an SDO_GEO_ADDR object.
SDO_GCDR.GEOCODE_ADDR_ALL	Geocodes an input address using attributes in an SDO_GEO_ADDR object, and returns matching addresses as an SDO_ADDR_ARRAY object.
SDO_GCDR.GEOCODE_ALL	Geocodes all addresses associated with an unformatted address and returns the result as an SDO_ADDR_ARRAY object.
SDO_GCDR.GEOCODE_AS_GEOMETRY	Geocodes an unformatted address and returns an SDO_GEOMETRY object.
SDO_GCDR.REVERSE_GEOCODE	Reverse geocodes a location, specified by its spatial geometry object and country, and returns an SDO_GEO_ADDR object.

3.1.6 Geocoding From a Place Name or Point-of-Interest (POI)

The name of a place or POI can be used in geocoding instead of a street address. In Example 10, a PL/SQL function `create_addr_from_placenam`e is created to

construct an SDO_GEO_ADDR object from placename and country input parameters. The `create_addr_from_placename` function is then used in a SELECT statement as input to the SDO_GCDR.GEOCODE_ADDR function, which will return a geocoded result for the place name and country specified.

Example 10: Geocoding from a Place Name

```
CREATE OR REPLACE FUNCTION create_addr_from_placename(
placename IN VARCHAR2,
country IN VARCHAR2)
RETURN SDO_GEO_ADDR
AS
addr SDO_GEO_ADDR ;
BEGIN
  addr := SDO_GEO_ADDR() ; -- construct empty address object
  addr.COUNTRY := country ;
  addr.PLACENAME := placename ;
  addr.MATCHMODE := 'DEFAULT' ;
  RETURN addr ;
END;
/

SELECT sdo_gcdr.geocode_addr('GEOCODER_US',
  create_addr_from_placename('CALIFORNIA PACIFIC MEDICAL CTR', 'US'))
FROM DUAL;
```

To improve the performance of the `create_addr_from_placename` function in Example 10, additional parameters, such as settlement, region, and postal code can be added to the input. Example 10.1 shows an updated version of the `create_addr_from_placename` function that accepts these additional parameters. To call this version of the function, the placename and country parameters must be specified, however, the other parameters can be NULL if their values are not known.

Example 10.1: Geocoding from a Place Name with Additional Parameters

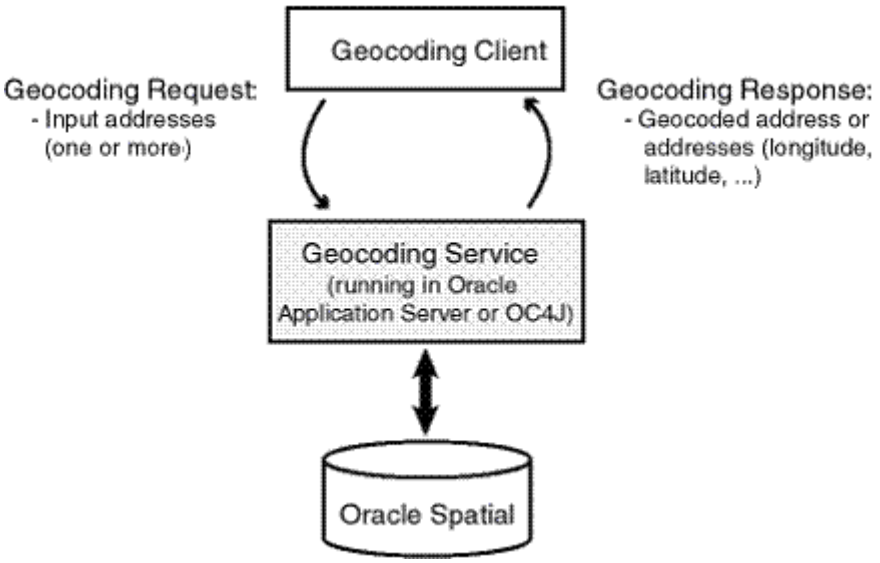
```
CREATE OR REPLACE FUNCTION create_addr_from_placename(
placename IN VARCHAR2,
city IN VARCHAR2,
state IN VARCHAR2,
postalcode IN VARCHAR2,
country IN VARCHAR2)
RETURN SDO_GEO_ADDR
AS
addr SDO_GEO_ADDR ;
BEGIN
  addr := SDO_GEO_ADDR() ; -- construct empty address object
  addr.SETTLEMENT := city ;
  addr.REGION := state ;
  addr.POSTALCODE := postalcode ;
  addr.COUNTRY := country ;
  addr.PLACENAME := placename ;
  addr.MATCHMODE := 'DEFAULT' ;
  RETURN addr ;
END;
/

SELECT sdo_gcdr.geocode_addr('GEOCODER_US',
  create_addr_from_placename('CALIFORNIA PACIFIC MEDICAL CTR',
    'san francisco', 'ca', NULL, 'US')) FROM DUAL;
```

3.2 Using the J2EE Geocoder

The J2EE geocoder is an XML API for a geocoding service. A client application communicates with the geocoding service via the HTTP protocol over the Internet: The client application sends an XML geocoding request containing a single input address or multiple input addresses to be geocoded. The geocoding service parses the geocoding request and invokes the geocoder to lookup the input address information in the database. It then sends the geocoded response in XML format to the client application. Figure 1, illustrates the flow of a geocoding request using the J2EE geocoder.

Figure 1: Geocoding Request Flow Using the J2EE Geocoder



After loading the geocoder schema into the database, the J2EE geocoder must be configured before it can be used.

3.2.1 Deploying and Configuring the J2EE Geocoder

The J2EE geocoder processes geocoding requests and generates responses. To enable this geocoding service, the geocoder.ear file (in $ORACLE_HOME/md/jlib) must be deployed using the Oracle WebLogic Server, Oracle Application Server (OracleAS) or a standalone installation of Oracle Application Server Containers for J2EE (OC4J). To deploy and configure the geocoding service, perform the appropriate step 1 and then follow the other steps 2-4:

Using Oracle Application Server or OC4J

1. Deploy the `geocoder.ear` file found in your $ORACLE_HOME/md/jlib directory using the OracleAS or the standalone OC4J Application Server Control (for example, http://<hostname>:8888/em). You may choose to deploy the geocoder.ear file in an existing OC4J instance, or you can create a new OC4J instance for the geocoder. In either case, enter **`geocoder`** for the *Application Name* during deployment.
   Note: If you are not familiar with application deployment using either OracleAS

or standalone OC4J, review the *Oracle Application Server Administrator's Guide* or the *Oracle Containers for J2EE Configuration and Administration Guide*.

Using Oracle WebLogic Server

1. The `geocoder.ear` file found in your $ORACLE_HOME/md/jlib directory must be unpacked before it can be deployed using the Oracle WebLogic Server (WLS). You will need to rename the geocoder.ear file and unpack its contents into a directory called **`../geocoder.ear`** . Rename the `web.war` file now found under the `$geocoder.ear/` directory and unpack its contents into a subdirectory called **`../web.war`** . Your directory structure should therefore be `$geocoder.ear/web.war/` . After unpacking the geocoder.ear and web.war files, copy the `xmlparserv2.jar` file found in your $ORACLE_HOME/LIB/ directory into the `$geocoder.ear/web.war/WEB-INF/lib/` directory. To deploy the `geocoder.ear`, logon to the WLS console (for example, http://<hostname>:7001/console) and from *Deployments*, *Install* the geocoder.ear accepting the name **`geocoder`** for the deployment and choosing the option to make the deployment accessible from a specified location.

2. Launch the Oracle Geocoder welcome page in a Web browser using the URL: `http://<hostname>:<port>/geocoder` . On the welcome page, select the `Administration` link and enter the admin (`oc4jadmin or weblogic)` username and password.

   Note: If you are using WLS and you are not using the default `weblogic` admin username, you will need to edit the `weblogic.xml` file located in the `$geocoder.ear/web.war/WEB-INF/directory`. Replace `<principal-name>weblogic</principal-name>`, with your WLS admin username, for example, `<principal-name>`**`my_weblogic_admin`**`</principal-name>.`

3. `Modify` the *Geocoder configuration file*. Uncomment at least one `<geocoder>` element and alter the `<database>` element attributes of that `<geocoder>` element to reflect the configuration of your database.

   Each <geocoder> element defines the geocoder for the database in which the geocoder schema resides. The <database> element defines the database connection for the geocoder. In Oracle 11g or later, there are two ways to define a database connection: 1) by providing the JDBC database connection parameters and 2) by providing the JNDI name (container_ds) of a predefined container data source. The attributes of the <database> element are as follows:

   `name`: a descriptive name for the database connection; it is not used to connect to the database
   `host`: the name or IP address of the database server
   `port`: the port number of the Oracle listener
   `sid`: the Oracle system identifier or Oracle Service name

`mode`: the type of JDBC driver to use for the connection
`user`: the database user under whose schema the geocoder schema is stored
`password`: the password for the user
`load_db_parser_profiles`: a parameter that specifies whether to load the address parser-profiles from the specified database connection. If true, the address parser-profiles are loaded from the geocoder schema, otherwise the parser profiles are loaded from the application at ../applications/geocoder/web/WEB-INF/parser_profiles/country_name.ppr (e.g. usa.ppr). Prior to Oracle 11g, parser-profiles were loaded from the application only. This parameter should be set to `true`
`container_ds`: The JNDI name for a predefined data source

Example 11, illustrates two different ways in which a <database> element can be defined. The first definition is a JDBC connection example, the second definition uses the JNDI name of a predefined container data source.

Example 11: <database> Element Definitions

```
<database name="gcdatabase"
          host="gisserver.us.oracle.com"
          port="1521"
          sid="orcl"
          mode="thin"
          user="geocoder_us"
          password="geocoder_us"
          load_db_parser_profiles="true" />

<database container_ds="jdbc/gc_europe"
          load_db_parser_profiles="true" />
```

Save changes after the configuration and then restart the geocoder.

Note: If the welcome page was not displayed at the beginning of this step, ensure that the newly deployed geocoding service was successfully started. It is assumed that you are running WLS 10.3.1.0 or later with an Oracle 11gR2 or later geocoder.ear file; or that you are running Oracle AS or OC4J 10.1.3 or later with an Oracle 11g or later geocoder.ear file.

4. To test the database connection, you may revisit the welcome page, *Oracle Spatial International Geocoder* at URL: `http://<hostname>:<port>/geocoder` and run the *International postal address parsing/geocoding demo* and the *XML geocoding request page*. These demos require geocoder data for the US.

3.2.2 XML Schema Definitions and Request and Response Examples

Geocoding Request Schema and Example

Request XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- Schema for an XML geocoding request that takes one or more input_locations
and supports reverse geocoding using the input_location's attributes -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:complexType name="address_lineType">
    <xsd:attribute name="value" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="address_listType">
    <xsd:sequence>
      <xsd:element name="input_location" type="input_locationType"
         maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="gdf_formType">
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="street" type="xsd:string"/>
    <xsd:attribute name="intersecting_street" type="xsd:string"/>
    <xsd:attribute name="builtup_area" type="xsd:string"/>
    <xsd:attribute name="order8_area" type="xsd:string"/>
    <xsd:attribute name="order2_area" type="xsd:string"/>
    <xsd:attribute name="order1_area" type="xsd:string"/>
    <xsd:attribute name="country" type="xsd:string"/>
    <xsd:attribute name="postal_code" type="xsd:string"/>
    <xsd:attribute name="postal_addon_code" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="gen_formType">
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="street" type="xsd:string"/>
    <xsd:attribute name="intersecting_street" type="xsd:string"/>
    <xsd:attribute name="sub_area" type="xsd:string"/>
    <xsd:attribute name="city" type="xsd:string"/>
    <xsd:attribute name="region" type="xsd:string"/>
    <xsd:attribute name="country" type="xsd:string"/>
    <xsd:attribute name="postal_code" type="xsd:string"/>
    <xsd:attribute name="postal_addon_code" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="geocode_request">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="address_list" type="address_listType"/>
      </xsd:sequence>
      <xsd:attribute name="vendor" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="input_addressType">
    <xsd:choice>
      <xsd:element name="us_form1" type="us_form1Type"/>
      <xsd:element name="us_form2" type="us_form2Type"/>
      <xsd:element name="gdf_form" type="gdf_formType"/>
      <xsd:element name="gen_form" type="gen_formType"/>
      <xsd:element name="unformatted" type="unformattedType"/>
    </xsd:choice>
    <xsd:attribute name="match_mode" default="relax_postal_code">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="exact"/>
          <xsd:enumeration value="relax_street_type"/>
          <xsd:enumeration value="relax_poi_name"/>
          <xsd:enumeration value="relax_house_number"/>
          <xsd:enumeration value="relax_base_name"/>
          <xsd:enumeration value="relax_postal_code"/>
          <xsd:enumeration value="relax_builtup_area"/>
          <xsd:enumeration value="relax_all"/>
```

```
          <xsd:enumeration value="DEFAULT"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="input_locationType">
    <xsd:sequence>
      <xsd:element name="input_address" type="input_addressType"
         minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string"/>
    <xsd:attribute name="country" type="xsd:string"/>
    <xsd:attribute name="longitude" type="xsd:string"/>
    <xsd:attribute name="latitude" type="xsd:string"/>
    <xsd:attribute name="x" type="xsd:string"/>
    <xsd:attribute name="y" type="xsd:string"/>
    <xsd:attribute name="srid" type="xsd:string"/>
    <xsd:attribute name="multimatch_number" type="xsd:string" default="1000"/>
  </xsd:complexType>
  <xsd:complexType name="unformattedType">
    <xsd:sequence>
      <xsd:element name="address_line" type="address_lineType"
         maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="country" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="us_form1Type">
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="street" type="xsd:string"/>
    <xsd:attribute name="intersecting_street" type="xsd:string"/>
    <xsd:attribute name="lastline" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="us_form2Type">
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="street" type="xsd:string"/>
    <xsd:attribute name="intersecting_street" type="xsd:string"/>
    <xsd:attribute name="city" type="xsd:string"/>
    <xsd:attribute name="state" type="xsd:string"/>
    <xsd:attribute name="zip_code" type="xsd:string"/>
  </xsd:complexType>
</xsd:schema>
```

XML Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<geocode_request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../geocode_request.xsd">
  <address_list>
    <input_location id="1">
      <input_address>
        <us_form2 name="Oracle" street="500 Oracle Parkway" city="Redwood City"
           state="CA" zip_code="94021"/>
      </input_address>
    </input_location>
    <input_location id="2">
      <input_address>
       <gdf_form street="1 Oracle Drive" builtup_area="Nashua" order1_area="NH"
       postal_code="03062" country="US"/>
      </input_address>
    </input_location>
    <input_location id="3">
      <input_address>
       <gen_form street="1 Oracle Drive" city="Nashua" region="NH"
       postal_code="03062" country="US"/>
```

```
      </input_address>
    </input_location>
    <input_location id="4">
      <input_address>
        <unformatted country="UNITED STATES">
          <address_line value="Oracle NEDC"/>
          <address_line value="1 Oracle drive "/>
          <address_line value="Nashua "/>
          <address_line value="NH"/>
        </unformatted>
      </input_address>
    </input_location>
  </address_list>
</geocode_request>
```

Geocoding Response Schema and Example

Response XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Schema for an XML geocoding response -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:complexType name="geocodeType">
    <xsd:sequence>
      <xsd:element name="match" type="matchType" minOccurs="0"
         maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="match_count" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="geocode_response">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="geocode" type="geocodeType" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="matchType">
    <xsd:sequence>
      <xsd:element name="output_address" type="output_addressType"/>
    </xsd:sequence>
    <xsd:attribute name="sequence" type="xsd:string" use="required"/>
    <xsd:attribute name="longitude" type="xsd:string" use="required"/>
    <xsd:attribute name="latitude" type="xsd:string" use="required"/>
    <xsd:attribute name="match_code" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="0"/>
          <xsd:enumeration value="1"/>
          <xsd:enumeration value="2"/>
          <xsd:enumeration value="3"/>
          <xsd:enumeration value="4"/>
          <xsd:enumeration value="10"/>
          <xsd:enumeration value="11"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="error_message" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="output_addressType">
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="house_number" type="xsd:string"/>
    <xsd:attribute name="street" type="xsd:string"/>
```

```
    <xsd:attribute name="builtup_area" type="xsd:string"/>
    <xsd:attribute name="order1_area" type="xsd:string"/>
    <xsd:attribute name="order8_area" type="xsd:string"/>
    <xsd:attribute name="country" type="xsd:string"/>
    <xsd:attribute name="postal_code" type="xsd:string"/>
    <xsd:attribute name="postal_addon_code" type="xsd:string"/>
    <xsd:attribute name="side" type="xsd:string"/>
    <xsd:attribute name="percent" type="xsd:string"/>
    <xsd:attribute name="edge_id" type="xsd:string"/>
  </xsd:complexType>
</xsd:schema>
```

XML Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<geocode_response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="../geocode_response.xsd">
 <geocode id="1" match_count="1">
    <match sequence="0"
         longitude="-122.26193971893862" latitude="37.53195483966782"
         match_code="10" error_message="????#ENUT?B281C??">
      <output_address name="" house_number="500" street="ORACLE PKY"
           builtup_area="REDWOOD CITY" order1_area="CA" order8_area=""
           country="US" postal_code="94065" postal_addon_code="" side="L"
           percent="0.33166666666666667" edge_id="28503563"/>
    </match>
 </geocode>
 <geocode id="2" match_count="1">
   <match sequence="0"
        longitude="-71.45937299307225" latitude="42.70784494226865"
        match_code="1" error_message="????#ENUT?B281CP?">
      <output_address name="" house_number="1" street="ORACLE DR"
           builtup_area="NASHUA" order1_area="NH" order8_area=""
           country="US" postal_code="03062" postal_addon_code="" side="L"
           percent="0.01" edge_id="22325991"/>
   </match>
 </geocode>
 <geocode id="3" match_count="1">
   <match sequence="0"
         longitude="-71.45937299307225" latitude="42.70784494226865"
         match_code="1" error_message="????#ENUT?B281CP?">
      <output_address name="" house_number="1" street="ORACLE DR"
           builtup_area="NASHUA" order1_area="NH" order8_area=""
           country="US" postal_code="03062" postal_addon_code="" side="L"
           percent="0.01" edge_id="22325991"/>
   </match>
 </geocode>
 <geocode id="4" match_count="1">
   <match sequence="0"
         longitude="-71.45937299307225" latitude="42.70784494226865"
         match_code="1" error_message="????#ENUT?B281CP?">
     <output_address name="" house_number="1" street="ORACLE DR"
           builtup_area="NASHUA" order1_area="NH" order8_area=""
           country="US" postal_code="03062" postal_addon_code="" side="L"
           percent="0.01" edge_id="22325991"/>
   </match>
 </geocode>
</geocode_response>
```

3.3 Using the Thin-Client Geocoder

The Thin-Client geocoder is a client-side Java API for geocoding. The Java API includes two Java classes, the *oracle.spatial.geocoder.client.GeocoderAddress* and the

oracle.spatial.geocoder.client.ThinClientGeocoder. Refer to the relevant Javadoc for the methods of these classes.

Note: The Thin-Client Geocoder is provided as sample code and is not intended for use in production systems. It is not supported by Oracle.

3.3.1 GeocoderAddress

The GeocoderAddress class is used to store address information, input addresses and geocoded result addresses. It also contains non-address attributes, such as, match mode, address ID and the maximum matched addresses that can be returned for each input address. Methods for manipulating these attributes are provided within the class.

3.3.2 ThinClientGeocoder

The ThinClientGeocoder class maintains the database connection and provides methods to geocode a single input address or a batch of input addresses. These methods geocode addresses by accessing the server-side PL/SQL geocoding API.

3.3.3 Thin-Client Geocoder Example

The following example class ThinClientTest, illustrates how the ThinClientGeocoder can be used to geocode addresses.

```
//ThinClientTest.java
import oracle.spatial.geocoder.client.*;
import java.util.*;

public class ThinClientTest
{
  public ThinClientTest()
  {
    ThinClientGeocoder geocoder = null ;
    try
    {
    //Create a new ThinClientGeocoder instance connecting to a database server
      geocoder = new ThinClientGeocoder("gisserver", "1521", "orcl", "gc_us",
      "gc_us", "thin") ;
    }
    catch(Exception e)
    {
      e.printStackTrace();
      return ;
    }

    // Create a GeocoderAddress with a formatted address
    GeocoderAddress ga1 = new GeocoderAddress() ;
    ga1.setId(1);
    ga1.setMatchMode("EXACT");
    ga1.setCountry("US");
    ga1.setStreet("1 Oracle drive");
    ga1.setSettlement("Nashua");
    ga1.setRegion("NH");

    // Geocode a single address
    ArrayList results = null ;
    try
    {
      results = geocoder.geocode(ga1) ;
    }
```

```
  catch(Exception e)
  {
    e.printStackTrace();
    return ;
  }
  for (int k = 0; k < results.size(); k++)
  {
    GeocoderAddress l = (GeocoderAddress)results.get(k);
    System.out.println(l) ;
  }

  ArrayList batch = new ArrayList(10) ;
  batch.add(ga1) ;

  // Create a GeocoderAddress with an unformatted address
  GeocoderAddress ga2 = new GeocoderAddress() ;
  ga2.setId(2);
  ga2.setMatchMode("DEFAULT");
  ga2.setCountry("US");
  String[] addressLines = new String[2] ;
  addressLines[0] = "1 oracce drive" ;
  addressLines[1] = "nashua, nh" ;
  ga2.setUnformattedAddressLines(addressLines);
  batch.add(ga2) ;

  results = null ;
  // Geocode a batch of addresses
  try
  {
    results = geocoder.batchGeocode(batch) ;
  }
  catch(Exception e)
  {
    e.printStackTrace();
    return ;
  }

  for (int k = 0; k < results.size(); k++)
  {
    ArrayList list = (ArrayList)results.get(k) ;
    for(int i=0; i<list.size(); i++)
    {
      GeocoderAddress res = (GeocoderAddress)list.get(i);
      System.out.println(res) ;
      if(res.isExactMatch())
        System.out.println("Exactly matched!") ;
      else
      {
        System.out.println("Not exactly matched!") ;
        System.out.println("House number matched:" +
                           res.houseNumberMatched());
        System.out.println("Street base name matched:" +
                           res.streetBaseNameMatched()) ;
        System.out.println("Street prefix matched:" +
                           res.streetPrefixMatched()) ;
        System.out.println("Street type matched:" + res.streetTypeMatched()) ;
        System.out.println("Street suffix matched:" +
                           res.streetSuffixMatched()) ;
        System.out.println("City matched:" + res.cityMatched()) ;
        System.out.println("Region matched:" + res.regionMatched()) ;
        System.out.println("Postal code matched:" + res.postalCodeMatched()) ;
      }
    }
  }
}

public static void main(String[] args)
{
```

```
    ThinClientTest sQLStubTest = new ThinClientTest();
  }
}
```

Oracle Spatial 11*g* Geocoder: An Oracle Technical White Paper
July 2011
Authors: Nicole Alexander and Ji Yang

Oracle Corporation World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.
Phone 650.506.7000
Fax 650.506.7200

International Inquiries:
Phone 44.932.872.020
Telex 851.927444(ORACLEG)
Fax 44.932.874.625

http://www.oracle.com