



Using EAN Database Geographical Regions

EAN Partner:Connect

Jon Arce

Overview

As part of our offerings to partners, EAN creates and maintains a full set of relational database files (34 in total), as well as custom scripts to automatically manage and update your local database file collection.

In this paper, we focus on the geography content of the files as per Expedia definitions of regions. For Expedia, a region, or more specifically a “marketing region,” is a bucket of hotels that represent a geography-related search term such as certain cities, airports, landmarks or neighborhoods. Think of them as places people will search in order to find hotels located within or nearby.

Technically, we will describe how to filter the records using the SQL language – in our case for the popular MySQL relational database, but the queries could be easily adapted for other databases.

Let's start by describing the relationship between the different Expedia regions.

EAN Regions

EAN defines a region as a container, a simple “bucket” that holds a collection of `EANHotelIDs`. These “buckets” are all related to geographical destinations, falling into these categories:

- Transportation – airports, train stations, ports of sail, etc.
- Cities – Cities, neighborhoods, multi-city areas, greater metro areas, etc.
- Point Of Interest – stadiums, landmarks, beaches, monuments, museums, etc.

So we will have a “bucket” of hotels for an `AirportCode` like JFK, a “bucket” for hotels near the Statue of Liberty, a bucket of hotels for “North Manhattan” as well as a bucket for the entire city of New York. From these examples, you can see that a hotel can belong to multiple regions at once.

These regions are in reality treated as **marketing regions**: they are NOT designed to be geographically correct. They will change to reflect new hotels that are available, for different weather seasons, high/low tourism seasons and even during special events like the Olympics. As an example, the region of Chicago, Illinois, USA will change to offer different inventory during winter vs. summer.

In this document we will present all possible regions and their hierarchical categories/subcategories, but most importantly, you will learn the required database queries to use for managing region data.

How to use a Region for API Integration

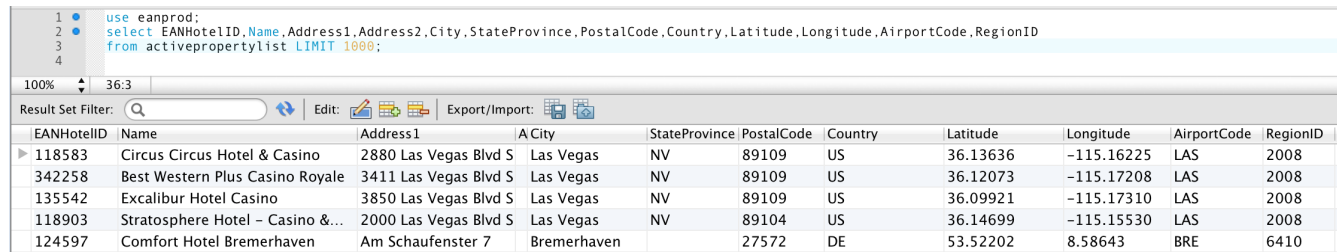
The usage of the region data is very straightforward - the geography is now in your hands! Here is a typical workflow:

1. Users search for the 'RegionName' - probably your typical autocomplete listing.
2. You resolve the 'RegionName' into a `RegionID` using our static database files.
3. From the `RegionID` you query for the list of `HotelIDs`.
4. Now into the EAN API to get availability, using the `getList` method with the list of `HotelIDs` as `<hotelIdList>`.

You also have the option of using the Center Coordinate (a GPS point) of the region, by using the EAN API getList method with <latitude> <longitude> searches. Let's start our overview of regions by looking into hotel properties themselves.

EAN Hotels

The ActivePropertyList static database file contains all the hotels in our inventory. There are various geography-related fields that you may wish to use, as you can see in this example:



The screenshot shows a database query interface. At the top, a SQL query is entered: `use eanprod; select EANHotelID, Name, Address1, Address2, City, StateProvince, PostalCode, Country, Latitude, Longitude, AirportCode, RegionID from activepropertylist LIMIT 1000;`. Below the query, a table of results is displayed with 11 columns: EANHotelID, Name, Address1, A City, StateProvince, PostalCode, Country, Latitude, Longitude, AirportCode, and RegionID. The first six rows of data are visible, showing hotels in Las Vegas and Bremerhaven.

EANHotelID	Name	Address1	A City	StateProvince	PostalCode	Country	Latitude	Longitude	AirportCode	RegionID
118583	Circus Circus Hotel & Casino	2880 Las Vegas Blvd S	Las Vegas	NV	89109	US	36.13636	-115.16225	LAS	2008
342258	Best Western Plus Casino Royale	3411 Las Vegas Blvd S	Las Vegas	NV	89109	US	36.12073	-115.17208	LAS	2008
135542	Excalibur Hotel Casino	3850 Las Vegas Blvd S	Las Vegas	NV	89109	US	36.09921	-115.17310	LAS	2008
118903	Stratosphere Hotel - Casino &...	2000 Las Vegas Blvd S	Las Vegas	NV	89104	US	36.14699	-115.15530	LAS	2008
124597	Comfort Hotel Bremerhaven	Am Schaufenster 7	Bremerhaven		27572	DE	53.52202	8.58643	BRE	6410

While this looks like the easiest way to map hotels to geography, we strongly advise you to NOT to use any of these fields to solve for geography. This is data collected and maintained directly by the hotels, not by EAN. Here are some specific reasons why using this data for geography **not recommended**:

1. **Hotel address** - Hotels normally will use what is called their 'vanity address' (and address that is somehow favorable for a traveler to select it). A classic example is hotels using 'Orlando, FL' while they are really in another neighboring city.
2. **PostalCode** - Some hotels use their mail postal code instead of the code for their physical location, or change it to match their 'vanity address'. In addition, the codes strings are not normalized, so we have different usages of uppercase/lowercase letters and separator characters.
3. **AirportCode** - The IATA AirportCode answers the question of “*If someone arrives from the USA, to your hotel, what Airport they most likely use?*”. This is unreliable as there are many hotels that service regional airports that do not directly receive USA traffic and could be more relevant. You could make a distance calculations by the GPS point of the hotel vs the airport to validate proximity.
4. **Latitude, Longitude** - The GPS point location of the hotel. We are improving on quality every day, as we use them to display the property on a map. We have created specific stored procedures and documentation on how to search by GPS location, please check our whitepaper: [Using EAN Databases for Geographical Solutions](#).
5. **RegionID** - This field value represents the main EAN Region, but it is NOT the only one where a hotel belongs. As a preference, Expedia maps them to a Multi-City (and Vicinity) type.

We will now clarify the proper way of mapping properties based on their EANRegionID associations.

EAN Region (City Tables)

The most basic way of mapping hotels is to use their parent city. We have two tables that represent a list of cities in our database files, which are:

1. **CityCoordinatesList** – Lists the cities and the coordinates that define the region over a map. Remember, our regions do NOT intend to match city definitions (but match it close-by). This is mainly used to create map overlays to allow click-and-search functionality, or when you need to display the geographical area where that hotel is located.
2. **RegionCenterCoordinatesList** – Lists the CenterLatitude and CenterLongitude of the particular region, it could be used to calculate distance in relation to any given point. Be careful, as sometimes the Center is considered the downtown of a city and it is NOT the mathematical center of the region. It is best used when searching for hotels close to the region center by GPS location. Please see our whitepaper: [Using EAN Databases for Geographical Solutions](#).
3. **ParentRegionList** – This table includes all regions, and allows you to determine the Parent of any given region. Regions begin at the country level, down into state-provinces, then into cities. It does include a lot of variants and will be properly explained in our next section. This is the main table to use to get all possible regions.

EAN Region (ParentRegionList)

The ParentRegionList table is the single most important source of geography information. If you do not need to draw a region, or know their polygon coordinates/center – if all you need is a list of locations – this is your table.

ParentRegionList also expresses the hierarchy of regions starting at the continent level, drilling down into the state-provinces and then into multi-regions, cities and neighborhoods. The names on this table are also available in multiple languages by downloading additional files (covered later in this paper).

Let's start by looking at an example of a hierarchy drill down:

Region ID	RegionType	Name	Parent ID
500001	Continent	North America	0 (nothing above this level)
201	Country	United States of America	500001
230	Province (State)	Nevada	201
6054173	Multi-City (within a country)	Clark County – Las Vegas	230
178276	Multi-City (Vicinity)	Las Vegas (and vicinity)	6054173
2008	City	Las Vegas	178276
800045	Neighborhood	The Strip (Las Vegas)	178276
6232223	Point of Interest Shadow	Welcome to Fabulous Las Vegas Sign	2008
6232224	Point of Interest	Welcome to Fabulous Las Vegas Sign	6232223

Let's clarify that this table is just a representation of the data and NOT the full possible targets of a drill down. For example, at the multi-region (within a country) level for 'Nevada' there are the following child regions: 'Clark County – Las Vegas', 'Central Nevada', 'Western Nevada', 'Northern Nevada – Humboldt River' and 'South Nevada'. There are many other cities inside the 'Clark County – Las Vegas' definition as well.

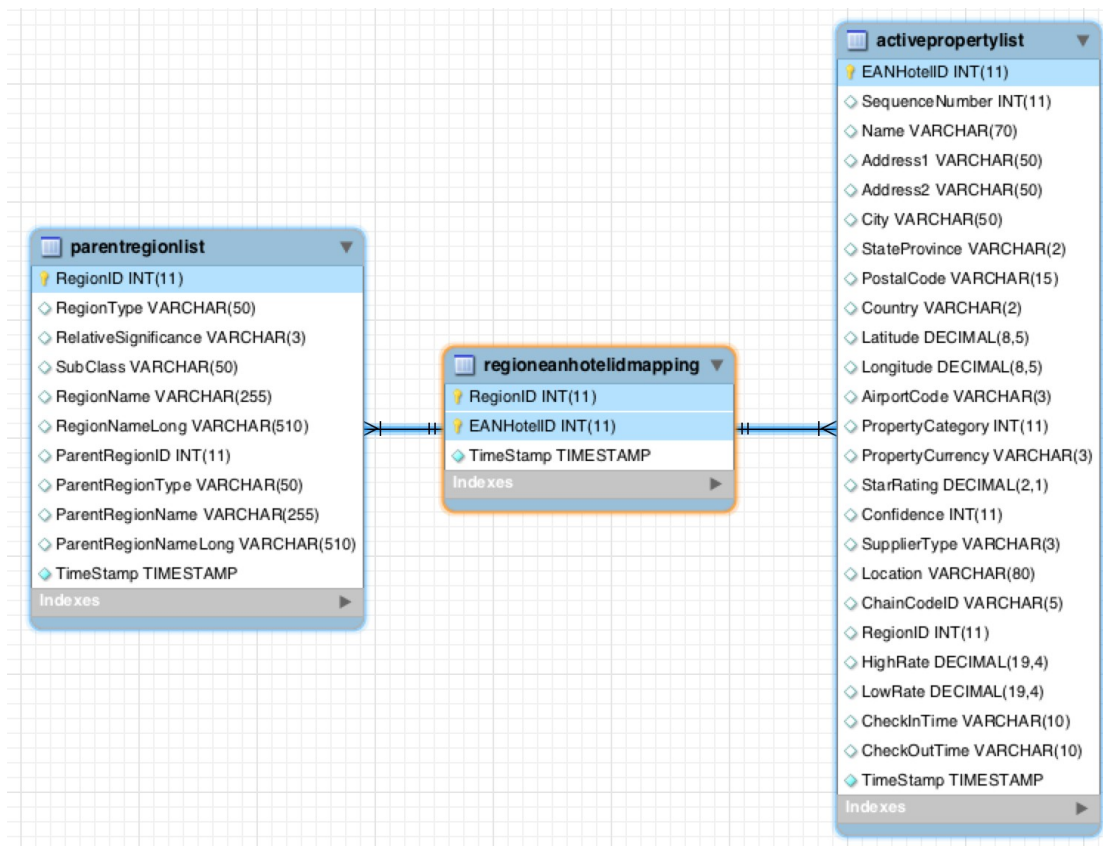
Remember, these are travel-related terms, not the proper political map/hierarchy of Nevada. If we were to follow this structure, in the case of USA, we would be based out of counties and then into cities. Remember this is NOT a geo-political system!

You may wonder, *can a city have a child region?* Yes, indeed: it could be a neighborhood name, but also we have points of interest (landmarks) that are inside that city. However, you will first get the “Point of Interest Shadow” types as a container for the true “Point of Interest”. There is no real usage for the “Shadow” other than it was necessary for the original relational database model to manage the information. We discuss a better way to get points of interest in their own section.

You could actually search at any given level and obtain a list of hotels, however as a general rule, you should start your searches at the city level, unless we are talking about small islands. Now, let's look at how to query for hotels.

EAN Hotels (link table)

Any given region will resolve into a list of hotels. However, any given hotel may belong to more than one region as well. In these cases of many-to-many relationships, we implemented a link table (also known as a join table) called `RegionEANHotelIDMapping`. This table joins `ParentRegionList` regions to a list of `EANHotelIDs` from `ActivePropertyList`. Here is the visual EER diagram of this relationship:



We could go the opposite direction too, starting from a hotel to find out how many regions it is included in. Let's use an hotel in 'Paris' (Hôtel Ares Eiffel) as an example:

RegionID	RegionName	EANHotelID	Name	RegionType	SubClass
2734	Paris	309567	Hôtel Ares Eiffel	City	
6187893	15th Arrondissement	309567	Hôtel Ares Eiffel	City	neighbor
6276077	Left Bank	309567	Hôtel Ares Eiffel	City	neighbor
179898	Paris (and vicinity)	309567	Hôtel Ares Eiffel	Multi-City (Vicinity)	
800093	Eiffel Tower – Orsay Museum (7 arr.)	309567	Hôtel Ares Eiffel	Neighborhood	regional
800101	Gare Montparnasse – Porte de Versailles (15 arr.)	309567	Hôtel Ares Eiffel	Neighborhood	regional
6002209	Palais Bourbon	309567	Hôtel Ares Eiffel	Point of Interest Shadow monument	
6002211	Palais de Chaillot	309567	Hôtel Ares Eiffel	Point of Interest Shadow sign	

108 row(s) returned.

You can see that this hotel belongs to 108 different regions! Let's now learn the different classifications for the regions.

EAN Region Types/SubClasses

There are various types of data that are defined in the `ParentRegionList` table, based on the groups created by the combinations of the fields `RegionType` and `SubClass`. The main `RegionTypes` (Continent, Country, Province (State)) do NOT have any `SubClass`.

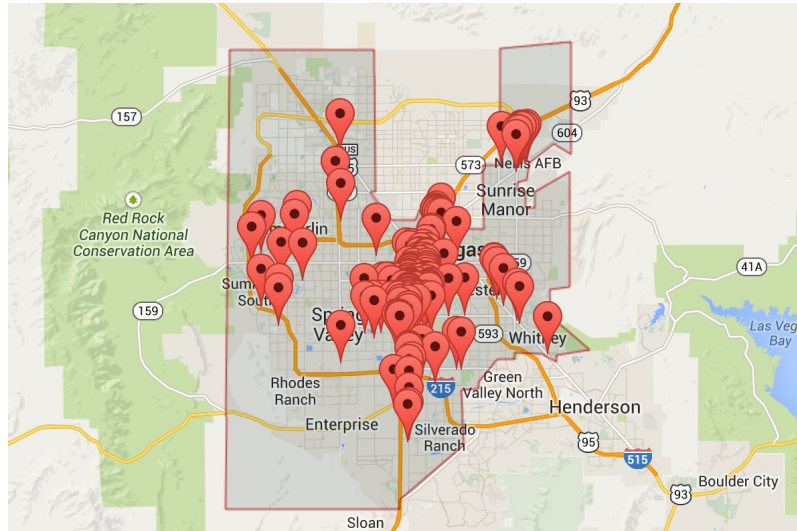
Let's review the possible groups. Below, we have included a P-index, consider it the Popularity Index (0 is highest) or how likely in our analysis we see people using those as search terms. Selecting a group is as easy as using the proper filter parameters in the SQL statement WHERE clause. For your convenience, we rate them as HIGH, MEDIUM, LOW relative to importance, and their Popularity (P## - 0 is highest), as a name may appear in different groups.

RegionType	SubClass	Remarks
City		Regular city names from the world. HIGH (51686 records) - P02
City	anchor, business, civic, historic, icecream, medical, monument, neighbor, regional, shopping, sign, skiing, stadium, sunglass, tree, winery	These records cover some Points Of Interest that are not in the usual sets with the same subclass (some unimportant) LOW (1728 records)
City	neighbor	Neighborhood names (not commonly used in the USA), districts, some villages and smaller Islands MEDIUM (12339 records) - P10
City	regional	Important well-known regions (e.g. Manhattan), islands, national parks and others. HIGH (384 records) - P05
Multi-City (Vicinity)		These areas include the city area and its surrounding areas, and also includes full Islands like 'Barbados (all)'. The list includes all famous cities. <u>This is what Expedia.com uses instead of a regular City record.</u> HIGH (2765 records) - P01

Multi-City (Vicinity)	district	To be applied in case of Regions that implements the (vicinity) concept
		LOW (2 records)
Multi-City (Vicinity)	regional	To be applied in case of Regions that implements the (vicinity) concept
		LOW (1 record)
Multi-Region (within a country)		This are regions that covers more than 1 city like Amazonas, BR – Andalucia, Spain or Bay Area, California.
		MEDIUM (1230 records) - P09
Multi-Region (within a country)	Autonomous community	Only one record Basque Country, Spain
		LOW (1 record)
Multi-Region (within a country)	region	Only one record Umbria, Italy
		MEDIUM (1 record) - P11
Neighborhood	airport	Better to use the Airport CoordinatesList table instead as this one only has 173 records.
		HIGH (1500 records) - P07
Neighborhood	city	Duplicated from the regular City types
		LOW (6232 records)
Neighborhood	downtown	Downtowns and City Centre areas
		HIGH (186 records) - P04
Neighborhood	anchor, civic, combo, golf, historic, icecream, medical, monument, school, shopping, sign, skiing, stadium, sunglass, theater, train, tree	All of those include records that are also represented as POIs
		LOW (331 records)
Neighborhood	neighbor	Neighborhoods and Zones, like: Cancun Hotel Zone, Central Park NYC, The Strip Las Vegas, Monte Carlo
		HIGH (1424 records) - P03
Neighborhood	regional	Islands, known zones like Broadway, East or West or South or North of some other region
		HIGH (730 records) - P06
Point of Interest Shadow	anchor, business, casino, civic, golf, historic, icecream, medical, monument, museums, school, shopping, sign, skiing, stadium, sunglass, theater, tree, winery	We consider POIs important, however depending on your focus you may choose to filter certain groups. Be careful as its size could impact autocomplete performance. <u>We prefer to use the PointOfInterestCoordinatesList table instead.</u>
		MEDIUM (89665 records) -P08

EAN Region (Cities)

We map cities to their more desirable locations and create a multi-polygons region. It is easier to understand if you visualize it; using Las Vegas example:



You can see that a city region is not a mere square over a city, but a hand-picked selection of regions that converts better. We aim for quality and not for quantity.

Query for Cities (technical)

To get a list of cities from our database you have two possible sources: using the `CityCoordinatesList` or filtering out the information from the `ParentRegionList`.

We prefer using the `ParentRegionList` table – let's start with a query just filtering out by `RegionType` like:

```
USE eanprod;
SELECT RegionID,RegionNameLong FROM parentregionlist
WHERE parentregionlist.RegionType IN('City');
```

RegionID	RegionNameLong
6023379	Région Test, Montréal, Canada
3433	Tirana, Albania
6047335	Durres, Albania
6052596	Sarande, Albania

66137 row(s) returned.

If you compare the number of results vs. the results of using the `CityCoordinatesList`, you'll notice both queries produce the same number of results. However in the `ParentRegionList` query, we are including all sub-types of the field `SubClass`. If we look deeper into the `SubClass` field (for `RegionType` that are `City`) we discover:

```
USE eanprod;
SELECT RegionID,RegionNameLong,RegionType,SubClass FROM parentregionlist
WHERE parentregionlist.RegionType='City' AND SubClass <> '';
```

RegionID	RegionNameLong	RegionType	SubClass
6160049	Wollaston Beach, Quincy, Massachusetts, United States of America	City	sunglass
6160076	Cleveland Circle, Boston, Massachusetts, United States of America	City	neighbor
6160599	Newbury Street, Boston, Massachusetts, United States of America	City	shopping
6161324	The Freedom Trail, Boston, Massachusetts, United States of America	City	historic

We get back SubClass field values such as:

- sunglass – a beach or beach-front location
- neighbor – neighborhood names (this set could be desirable to include)
- shopping – shopping malls, districts or famous shopping streets
- historic – historic trails or landmarks

We do NOT want those in our query! You should always include both a RegionType and a SubClass in your queries. We need then to refine the City query to be:

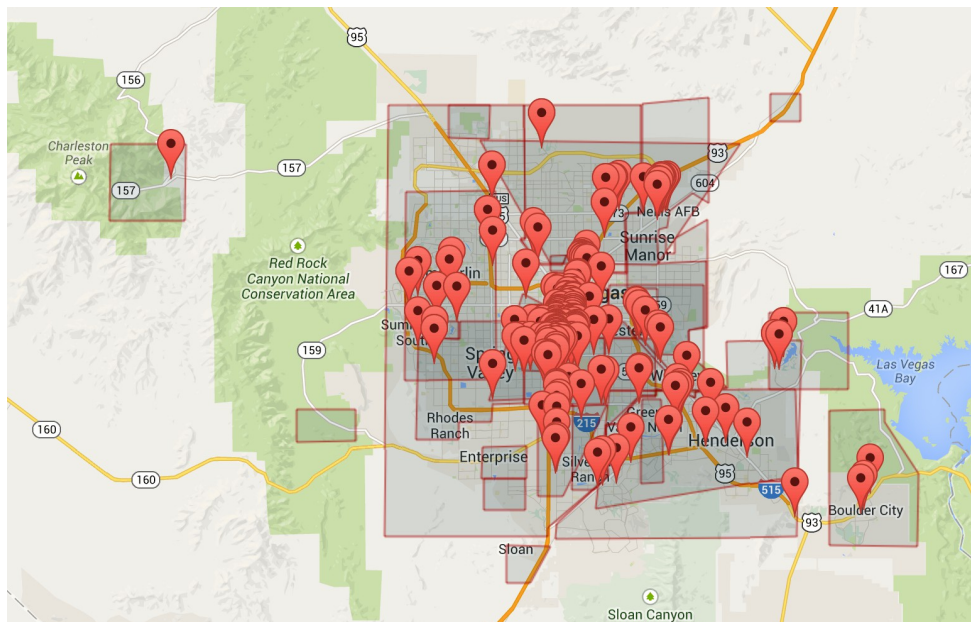
```
USE eanprod;  
SELECT RegionID, RegionNameLong, RegionType, SubClass FROM parentregionlist  
WHERE parentregionlist.RegionType='City' AND SubClass='';
```

This is our recommended way of getting the list of Cities. Always make sure you are using the proper types and classes value on your queries. The reason is that even if today we do not have SubClass values for certain RegionType, this may change in the future.

EAN Regions (query for Multi-City(and vicinity))

For most world-famous major city markets, such as Paris, Las Vegas, Orlando, Miami, London, Tokyo, etc., Expedia utilizes the powerful concept of **CityName (and vicinity)**. This is a special geographic region that covers the best possible areas inside the city and close to it, that in our analysis, customers prefer when searching for that city name. The idea is to present a mix of classics and off-path trendy properties. This regions normally convert more revenue than the plain CityName areas. If you want to follow the Expedia model, when customers look for the CityName, show instead results for the **CityName (and vicinity)** instead.

In our previous example we used just the city of Las Vegas, now let's look at the “Las Vegas (and vicinity)” region:



As you can see we cover a greater area than the explicit “Las Vegas” region from the previous section. It is possible that this coverage may not be desirable in all situations – it will be up to your particular integration to use or not use this type of area.

Query for Multi-City (Vicinity) (technical)

To get a list of CityName (and vicinity) from our database you will use once again the ParentRegionList table:

```
USE eanprod;  
SELECT RegionID,RegionNameLong,RegionType,SubClass FROM parentregionlist  
WHERE RegionType='Multi-City (Vicinity)' AND SubClass='';
```

With results:

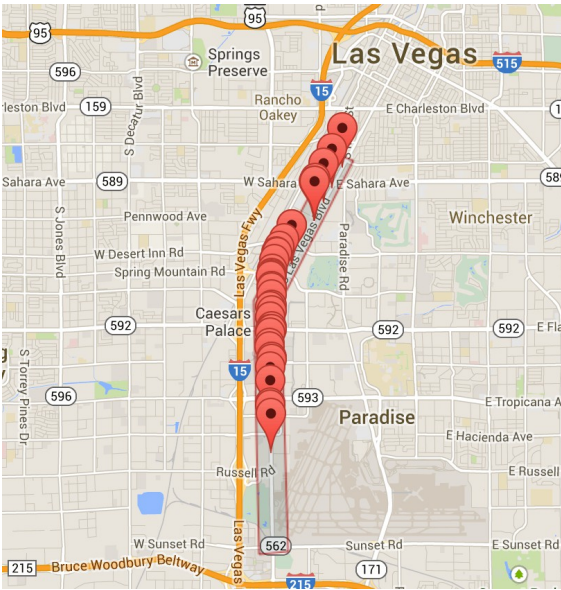
RegionID	RegionNameLong	RegionType
259	Aarhus (and vicinity), Denmark	Multi-City (Vicinity)
266	Abilene (and vicinity), Texas, United States of America	Multi-City (Vicinity)
277	Albany (and vicinity), Georgia, United States of America	Multi-City (Vicinity)
280	Accra (and vicinity), Ghana	Multi-City (Vicinity)
282	Lanzarote, Spain	Multi-City (Vicinity)
286	Waco (and vicinity), Texas, United States of America	Multi-City (Vicinity)

Notice that the (and vicinity) string is NOT always present in the results. We could use the created function of REGION_NAME_CLEAN to eliminate the (and vicinity) from the results strings.

If you need to filter by SubClass include it on the query, refer to the **EAN RegionTypes/SubClasses** section in this document for more information.

EAN Region (Neighborhood)

Neighborhood areas cover very specific regions that people tend to know and search for when traveling. These neighborhoods could be even as famous as the city, and maybe preferred over some lesser-known cities. For example if you analyze the Expedia site, you will notice that for the Las Vegas market, we present two options: “Las Vegas (and vicinity)” and “Las Vegas Strip”. Let's look now at neighborhood named “Las Vegas Strip” region:



This is a collection of hotels that are on the stretch called “Las Vegas Strip”.

Query for Neighborhood (technical)

To get a list of neighborhoods from our database, you will use once more the ParentRegionList table:

```
USE eanprod;
SELECT RegionID,RegionNameLong,RegionType,SubClass FROM parentregionlist
WHERE RegionType='Neighborhood' AND SubClass IN ('downtown','neighbor','regional');
```

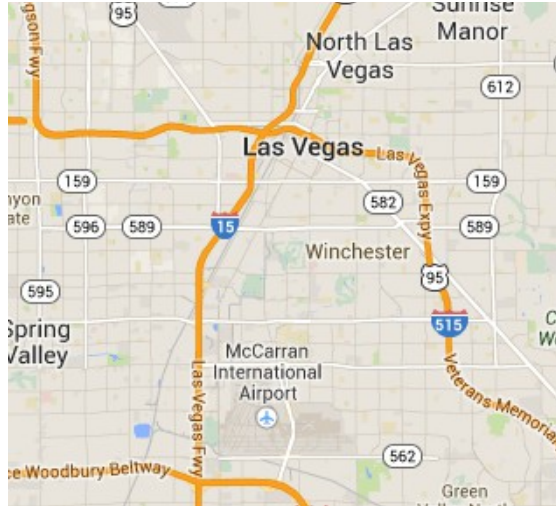
With results:

RegionID	RegionNameLong	RegionType	SubClass
506	Barbuda, Antigua and Barbuda	Neighborhood	regional
916	Carriacou Island, Grenada	Neighborhood	regional
1038	Dunhuang, Jiuquan, China	Neighborhood	neighbor
1116	Tortola, British Virgin Islands	Neighborhood	regional
1350	Green Island, Taiwan	Neighborhood	regional
1479	Hailar, Hulunbuir, China	Neighborhood	neighbor
1505	Ha'apai, Tonga	Neighborhood	regional
2049	Lands End, England, United Kingdom	Neighborhood	regional

Notice that we use the SubClass to get only certain classes of data. Refer to the **EAN RegionTypes/SubClasses** section in this document for more information.

EAN Region (Airports)

Airports are a special type of regions as they are NOT an overlay map representation of a geography. If we follow once again the Las Vegas example, the most popular airport being “McCarran International Airport” we can see (or not see):



There is NO overlay for an airport – the Airport are chosen by the hotel when we ask the question “*What is the main airport that service your customers?*”. In that sense, an airport region does not means properties that are close-by or surrounding the airport.

Query for Airports (technical)

To get a list of airports, we use the table `AirportCoordinatesList`, looking for “LAS” (the `AirportCode` is the IATA Code):

```
USE eanprod;  
SELECT * FROM airportcoordinateslist WHERE AirportCode="LAS";
```

Results:

AirportID	AirportCode	AirportName	Latitude	Longitude	MainCityID	CountryCode
6000345	LAS	Las Vegas, NV, United States (LAS-McCarran Intl.)	36.085390	-115.150100	178276	US

We can notice two ids present the `AirportID` and the `MainCityID`. We use the `MainCityID` and discover it is:

RegionID	RegionNameLong	RegionType	SubClass
178276	Las Vegas (and vicinity), Nevada, United States of America	Multi-City (Vicinity)	

Basically, the airports are priority-mapped into the `MainCity` (and `Vicinity`). This is the correct way of displaying results for the airport, in this case there will be 219 hotels serviced by the LAS airport.

Looking at a joined query with the `ParentRegionList` we retrieved earlier:

```
USE eanprod;  
SELECT MainCityID AS 'RegionID', AirportName, RegionNameLong FROM  
airportcoordinateslist
```

JOIN parentregionlist ON
airportcoordinateslist.MainCityID=parentregionlist.RegionID;

Results:

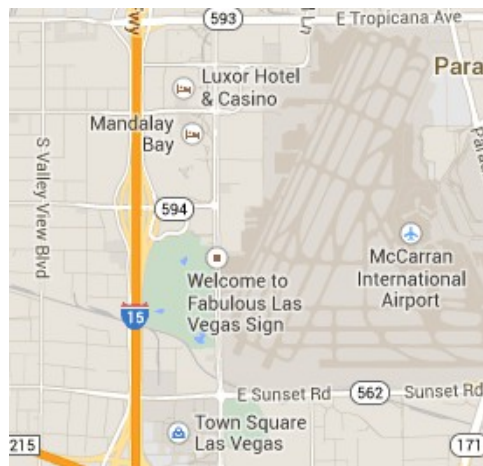
RegionID	AirportName	RegionNameLong
6058120	Albany, GA, United States (ABY–Southwest Georgia...	Tifton (and vicinity), Georgia, United States of America
278	Aberdeen, Scotland, UK (ABZ)	Aberdeen, Scotland, United Kingdom
279	Acapulco, Guerrero, Mexico (ACA–General Juan N....	Acapulco, Guerrero, Mexico
280	Accra, Ghana (ACC–Kotoka Intl.)	Accra (and vicinity), Ghana
282	Arrecife, Spain (ACE–Lanzarote)	Lanzarote, Spain

If we analyze this for cities where the airport is very far from the city center, like the case of London, you may prefer to **find hotels near the airport**. In these cases, you should use the Latitude and Longitude fields and either use the database stored procedure: sp_hotels_from_point or via the API using the getList by latitude/longitude method.

EAN (Point of Interest)

For points of interest, we use the concept of a **Point of Interest Shadow**. It may sound misleading, but think of it as “shadow” the POI creates, or simply as the area around the point of interest. To be crystal clear, you will search for the “Shadow” RegionID, not for the point of interest itself.

The shadow holds the RegionID that tells us the hotels that surround the POI. If we follow once again the Las Vegas example, let's search for the famous “Welcome to Fabulous Las Vegas” sign:



In the case of the sign, it is classified as a monument. There are many SubClassifications such as:

- **anchor** – marinas and ports (ex. Marina Del Rey, California) – 1507 records
- **business** – famous business buildings and locations (ex. Abbey Road Studios, London) – 75 records
- **casino** – gambling places & casinos (ex. Golden Nugget Casino in Las Vegas) – 470 records
- **civic** – cultural centers, halls, convention centers, libraries (NYC Library, New York City) - 4038 records
- **golf** – golf courses and country clubs (Pebble Beach Golf Course, California) – 4820 records
- **historic** – Villages, ruins, castles and similar (Royal Castle, Warsaw, Poland) – 6116 records
- **icecream** – funny name! But it covers park, zoo, amusement parks, aquariums – think of children's entertainment or family vacations (but not limited to!) (Disney World, Orlando, Florida, USA) – 5222 records
- **medical** – famous hospitals, (KEM Hospital, Mumbai, India) – 954 records
- **monument** - famous monuments (Eiffel Tower, Paris, France) – 14401 records

- **museums** – museums of all kinds (Bob Marley Museum, Kingston, Jamaica) – 12907 records
- **schools** – international universities and Colleges (University of Zaragoza, Zaragoza, Spain) – 2853 records
- **shopping** – shopping malls, districts and streets (Bond Street, London, England) – 4194 records
- **sign** – plazas, squares and signs (Welcome to Fabulous Las Vegas Nevada Sign) – 2452 records
- **skiing** – ski resort, ski Lift, ski area (Furano Ski Area, Asahikawa, Japan) – 1808 records
- **stadium** – stadiums, bowls and arenas (sport related) (San Siro Stadium, Milan, Italy) – 3719 records
- **sunglass** – beaches (Ipanema Beach, Rio de Janeiro, Brazil) – 5053 records
- **theater** – theaters, opera (Komische Oper Berlin, Berlin, Germany) – 3455 records
- **tree** – public parks, valleys, lakes, mountains (City Park, Biel, Switzerland) – 14305 records
- **winery** – wineries (Karikari Estate Winery, Matauri Bay, New Zealand) - 1316

You can use these SubClassifications to filter out the ones you need.

Query for POIs (technical)

To get a list of points of interest, we prefer to use the `PointOfInterestCoordinatesList`. In this table, we filter the `SubClass` on the field `SubClassification`. It also includes the latitude and longitude of the location that we could use to calculate the distance from the hotel (could use a Haversine formula or similar implementation). For example, searching for all stadiums would look like:

```
USE eanprod;
SELECT * FROM pointsofinterestcoordinateslist WHERE SubClassification="stadium";
```

Results:

RegionID	RegionName	RegionNameLong	Latitude	Longitude	SubClassification
6068000	Abdi Ipekci Arena	Abdi Ipekci Arena, Istanbul, Turkey	40.996970	28.920007	stadium
6113595	Abe Lenstra Stadium	Abe Lenstra Stadium, Heerenveen, Netherlands	52.959026	5.934537	stadium
6090604	ABSA Stadium	ABSA Stadium, Durban, South Africa	-29.824860	31.028489	stadium
6111537	Abuja Stadium	Abuja Stadium, Abuja, Nigeria	9.037935	7.453367	stadium
6201132	Addington Raceway	Addington Raceway, Christchurch, New Zealand	-43.544678	172.601315	stadium
6111933	Addis Ababa Stadium	Addis Ababa Stadium, Addis Ababa, Ethiopia	9.013294	38.756415	stadium
6085870	Adelaide Entertainment Centre	Adelaide Entertainment Centre, Adelaide, South Austr...	-34.907614	138.574161	stadium
6085834	Adelaide Oval	Adelaide Oval, Adelaide, South Australia, Australia	-34.915368	138.597493	stadium

Notice that the `RegionID` we get from this query will be the same as its `Point of Interest Shadow`, from the `ParentRegionList` table.

Region Names in other Languages

For each of the 34 languages that we support, there is a single file that includes the translations. These files use the name format `RegionList_xx_xx`, where last part of the name is the locale code. For example the spanish from Spain translation will be named: `RegionList_es_es`, for Brazilian Portuguese: `RegionList_pt_br`.

To use a different language, just add a new `JOIN` clause with an `ON` clause of the matching `RegionID`. For example to get the names of all regions in English, Spanish and Portuguese, we would do the following:

```
USE eanprod;
SELECT parentregionlist.RegionID, parentregionlist.RegionNameLong AS 'English',
regionlist_es_es.RegionNameLong AS 'Spanish', regionlist_pt_br.RegionNameLong AS
'Portuguese'
FROM parentregionlist
JOIN regionlist_es_es ON parentregionlist.RegionID = regionlist_es_es.RegionID
JOIN regionlist_pt_br ON parentregionlist.RegionID = regionlist_pt_br.RegionID
WHERE parentregionlist.RegionNameLong LIKE "%Las Vegas%";
```

Results will look like:

RegionID	English	Spanish	Portuguese
2008	Las Vegas, Nevada, United States of America	Las Vegas, Nevada, Estados Unidos	Las Vegas, Nevada, Estados Unidos da América
8605	Las Vegas, New Mexico, United States of America	Las Vegas, Nuevo México, Estados Unidos	Las Vegas, Novo México, Estados Unidos da América
9169	North Las Vegas, Nevada, United States of America	Norte de Las Vegas, Nevada, Estados Unidos	North Las Vegas, Nevada, Estados Unidos da América
74507	Arden, Las Vegas, Nevada, United States of America	Arden, Las Vegas, Nevada, Estados Unidos	Arden, Las Vegas, Nevada, Estados Unidos da América
76704	Bard, Las Vegas, Nevada, United States of America	Bard, Las Vegas, Nevada, Estados Unidos	Bard, Las Vegas, Nevada, Estados Unidos da América
82198	Bracken, Las Vegas, Nevada, United States of America	Bracken, Las Vegas, Nevada, Estados Unidos	Bracken, Las Vegas, Nevada, Estados Unidos da América
143393	Paradise, Las Vegas, Nevada, United States of America	Paradise, Las Vegas, Nevada, Estados Unidos	Paradise, Las Vegas, Nevada, Estados Unidos da América
165607	The Lakes, Las Vegas, Nevada, United States of America	The Lakes, Las Vegas, Nevada, Estados Unidos	The Lakes, Las Vegas, Nevada, Estados Unidos da América

You will need to create as many translations tables as languages you plan to support. Refer to our online documentation on how to create the multi-language tables.

Autocomplete

An autocomplete is the typical feature used on travel websites that allows the customer to begin typing and quickly see a possible list of 'complete' terms. We can see such functionality in our own Chameleon template platform:

Search for Hotels

The screenshot shows a search interface titled "Search for Hotels". On the left, there's a green sidebar with the text "Where would you like to go?" and a list of cities with radio buttons: Las Vegas, Chicago, San Diego, Los Angeles, Orlando, Atlanta, San Francisco, New Orleans, New York, London, Boston, and Washington. In the center, a search bar contains the text "Miami". Below the search bar, a dropdown menu is open, displaying a list of suggestions categorized by "Cities/Areas", "Airports", "Landmarks", and "Hotels". The suggestions include "Miami, FL, United States", "Miami Beach, Dania, FL, United States", "Miami (AREA), FL, United States", "Miami Gardens, Dania, FL, United States", "Miami Lakes, Dania, FL, United States", "Miami Shores, Dania, FL, United States", "Miami Springs, Miami, FL, United States", "Miami International Airport (MIA)", "Miami Public Seaplane Base Airport (MPB)", "Miami Beach, Charnocks, Barbados", "Miami Intl. Airport (MIA), Miami, FL, United States", "Miami Amtrak Station, Dania, FL, United States", "MIAMI BEACH RESORT, Dahab, EG", "Miami Beachside Holiday Apartments, Miami, Queensland, AU", "Miami, Jesolo, IT", and "Miami Hotel, Chiang Mai, TH". At the bottom of the dropdown, there's a prompt "...keep typing to refine your search". To the right of the dropdown, there are three small images: a city skyline, a beach, and a hotel building. Below the images, there are three tabs: "Miami Beach" with a price of "\$51", "New Orleans" with a price of "\$50", and a third tab that is partially visible.

You can see we have classifications like: **Cities / Areas, Airports, Landmarks, Hotels**. Using the suggested queries presented in this document, we could create a list based on RegionIDs that you could use to present this information. When a user makes a selection, we would use the **RegionID** to discover the list of **EANHotelIDs** to send to the EAN API within <hotelIdList>.

For our sample autocomplete, we create a table with fields like:

```

DROP TABLE IF EXISTS autocomplete;
CREATE TABLE autocomplete
(
    English VARCHAR(510),
    EANRegionID INT,
    Latitude NUMERIC(9,6),
    Longitude NUMERIC(9,6),
    EANHotelCount INT,
    EANHotelIDList TEXT,

```

```

    DisplayType VARCHAR(20),
    TimeStamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8_unicode_ci;

CREATE FULLTEXT INDEX ft_en ON destinations(English);

```

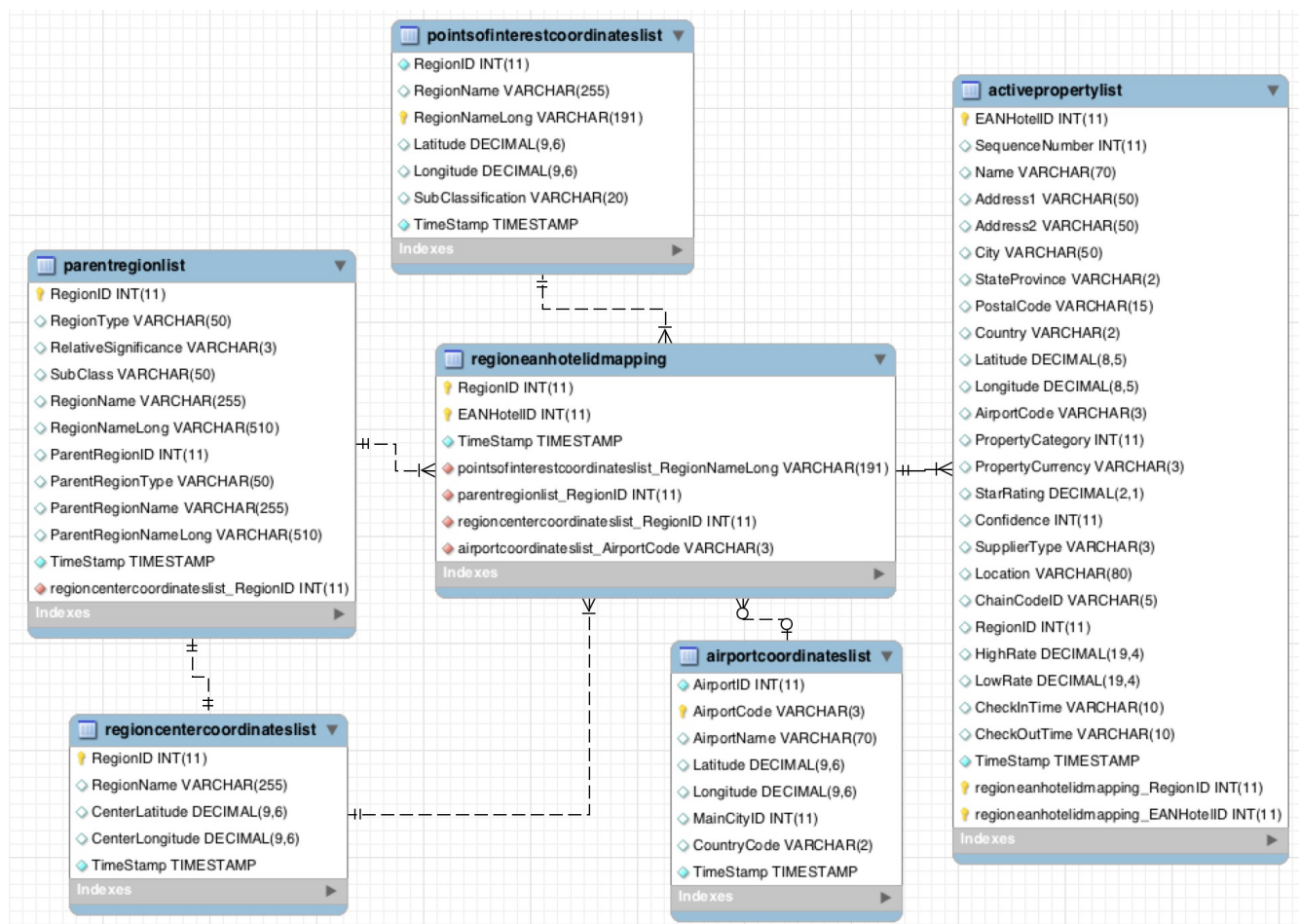
We will use the RegionNameLong in each of our languages and will save the full list of EANHotelIDs for each regions inside this table, just as a convenience. For that, we will use the TEXT data field (max. of 65535 characters) as we may get thousands of hotels for some destinations like London. We will also save the longitude and latitude, so we can always switch to search by the geopoint.

For the autocomplete we will be using six different tables:

1. ParentRegionList – to get the Cities, Multi-City (and vicinity) and Neighborhood data.
2. RegionCenterCoordinatesList – to get the GPS point (latitude and longitude) of the region.
3. AirportCoordinatesList – to get the airport data and its GPS point.
4. PointOfCoordinatesList – to get all POIs and their GPS points.
5. RegionEANHotelIDMapping – the join table to get the list of properties
6. ActivePropertyList – as we choose to include also hotel names in the autocomplete, this contains the name, EANHotelID and the GPS point

If we what to support more languages (like Spanish and Portuguese in our example) we will need to add 1x more table per language (regionlist_es_es and regionlist_pt_br).

As a diagram it will look like:



All that is needed is to create a big INSERT INTO query with each group of data. Due to its length, we do not include it directly in this document but you may download the example from our [github repository](#) as part of the eanextras database. The logic is very simplistic:

```
## Create the Table
# Insert Records from select * of (
## the Multi-City (and vicinity) records
UNION ALL
## the City records (includes the regional)
UNION ALL
## the Neighborhood records (downtown / neighbor / regional)
UNION ALL
## the Airports Records
UNION ALL
## the Point Of Interest Records
UNION ALL
## the Hotels Records
) WHERE EANHotelIDCount > 0;
```

We normalize the queries, and just UNION ALL of them together, then insert into the just created table ONLY if they have records in their Regions. In the case of MySQL we use the MYISAM engine to support the fast text search functionality needed for the autocomplete. This query produced around 256K records, but we filter ONLY those Regions where we have hotels to sell, taking it down to 234K including the hotels. Our records consist of:

DisplayType	Amt
Airports	2685
Cities/Areas	33352
Hotels	117218
Landmarks	81193

Depending on your market and customers, you may wish to adjust this setup to your needs - please do not take this as one-size fits all! In an autocomplete scenario, your query will be tied-up to the database. As people type, you will issue a query to return the data, such as:

```
USE eanextras;
SELECT EANRegionID AS 'Target' , English AS 'Name' , DisplayType AS 'Category'
FROM autocomplete WHERE English LIKE CONCAT('%',@searchterm,'%') LIMIT 30;
```

GUI libraries such as *jQuery Autocomplete* could easily do this type of task.

Database helper functions

As part of our downloadable scripts, we do create the full structure of the eanprod database, but there are also some functions that will help managing the geographical tables and its data. Let's see what they can do.

Database Function (REGION_NAME_CLEAN)

This function can be used to eliminate anything that is inside a string. We can use it to eliminate things like '(type 7)' or '(and vicinity)' from the Region Names. As an example:

```
SELECT eanprod.REGION_NAME_CLEAN('Orlando (and vicinity)');
```

With the expected results of:

Orlando

We use it to clean up the names of the City (and vicinity) types, so it shows clean in the autocomplete list.

Database Function (HOTELS_IN_REGION)

This function allows us to get a list of hotelIDs (comma delimited) for any RegionID number. We use it to avoid complicating the queries to join into the RegionEANHotelIDMapping. As an example:

```
#hotel list in The Strip, Las Vegas
SELECT eanprod.HOTELS_IN_REGION(800045);
```

With the expected results of:

```
107128,108540,111474,112914,113149,115163,115838,116531,118583,118903,119566,121569
,122212,123169,123792,124363,129410,134556,135542,147594,149069,161293,163430,16343
8,163446,174511,212291,228169,230758,259462,259607,259630,273081,273427,311298,3328
41,342258,364984,432875,469402
```

The idea here is to easily store this value. The function internally sets the session of the group_concat() database function to a high number to avoid truncating the list for destinations like Paris, France, where there are thousands of hotels. It is also very easy to use the results to pass into an API RESTful call as hotelIdList.

Database Function (HOTELS_IN_REGION_COUNT)

This function is similar to HOTELS_IN_REGION but instead returns the amount of hotels that would make the list of any RegionID number. We use it to save the data in the autocomplete table. Partners find it useful to limit regions that have little or no inventory. Data for this function comes from a query to RegionEANHotelIDMapping. As an example:

```
#amount of hotels in region The Strip, Las Vegas
SELECT eanprod.HOTELS_IN_REGION_COUNT(800045);
```

With the expected results of:

```
40
```

This value could also be used in WHERE clauses to limit or filter results.

Database Function (TRANSLITERATE)

When displaying other Latin-based languages like Italian, French, Spanish and Portuguese. You will find that the users sometimes type the correct accented characters while other times they will not. So a search for México should behave the same as Mexico.

You could manage this using the database sorting / search, but we prefer to manage on the search itself. The TRANSLITERATE function that we add to the database basically convert characters by position from this string:

- source string - 'ÁáãäåĖĖĕĕĕĭĩĩıŌőôôŮůüũñŇçç'
- destination string - 'AaaaaEEeeelliiOOooUuunNCC'

Notice that we DO NOT COVER all possible languages with just the string above, our advice will be to adjust the function to the transliteration characters that you need.

If you are aiming for the fastest possible conversion, we will advise to change the data in the table itself creating two pairs of names per language, something like:

- `display_name_es_es` – '*Ciudad de México*'
- `transliterated_name_es_es` – 'Ciudad de Mexico'

One advantage here is to add other spelling variations for the same destination, so we could change `transliterated_name_es_es` – 'Ciudad Mexico', 'Mexico D.F.', 'Mexico City' that will increase the matches to the search, but of course will complicate the search logic on your integration.

Other Geographic Tables

There are other geographical tables with variants of content that you may need for your integration. They are not considered main components, but for completeness, let's list them with possible usages:

- `AliasRegionList` – lists different spellings or names for the same region (however this table currently presents Alias in different languages, but wrongly identified, so its use may be limited). There are localized versions for other languages of this table.
- `CityCoordinatesList` – for each region it lists the geo-point that construct the polygon that defines the region. Useful if you need to illustrate over a region over a map.
- `CountryList` – list of all countries, with ISO country codes interesting if contains a transliteration of the name (non accented version). There are also localized versions of this table.
- `NeighborhoodCoordinatesList` – similar to `CityCoordinatesList` but for the Neighborhoods, it presents the polygon that describes the area. Only useful if you want to display the region over a map. In the `RegionName` field you will find a “(type #)” value – the number refers to:
 1. Neighborhood
 2. Downtown/City Center
 3. Secondary Cities – Cities close to bodies of water (Lake or Sea) however there are some very far, so please do not use as indicator of “near the water” (use the `CityCoordinatesList` instead)
 4. Airport Vicinity
 5. Regional names & others (like north, south, east, west), also covers amusement parks and others)
 6. Landmark / Place
 7. Landmark / Beach
 8. Train Station Vicinity
- `RegionCenterCoordinatesList` – Gives you the `CenterLatitude` and `CenterLongitude` geo-point of regions (excluding airports and Points of interests).

As you can see, these tables will be mostly useful for mapping (over a map service) and for displaying or solving alternate names of the regions.

Real Location (reverse geocoding)

Some people ask us if we maintain a list of 'real' address of hotels, landmarks and cities. As you can tell, we do not follow a 'precise' geo-political mapping of the world (think on ISO standardized regions). We understand there are multiple scenarios when that is desirable, such as:

- Multi-supplier scenarios – Trying to compare a hotel from one supplier to know if it is the same as another supplier.
- Map-based applications – using for example Google maps as a search or display template.
- Data re-hash – like combining multiple transportations (train, taxis) pre-calculated as supplemental information to travelers
- More precision on neighborhood information – For example in Latin America and also Spain & France, there are multiple neighborhood names for a place.
- Postal Codes – We do not include the postal codes on all address, or they are not normalize you may need this info. For maps.

In those case you are probably looking at what is known as **reverse geocoding**. Reverse geocoding is the process of taking a point location (latitude, longitude) and resolving it to a readable address or place name. This permits the identification of nearby street addresses, places, and/or areal subdivisions such as neighborhoods, county, state, or country.

To achieve this, you could use an online service (pay or free) or even create your own server to solve this information. Here is a list of some known APIs to do such operation:

- [Google Geocoding API](#) – paid after certain limit
- [Bing Maps API \(Locations API\)](#) – paid after certain limit
- [Yahoo! \(Boss Geo Services\)](#) – paid after certain limit
- [MapQuest Geocoding API](#) – paid after certain limit
- [Nokia Geocoder API](#) – paid after certain limit
- [OpenStreetMap \(Nominatim API\)](#) – free (but limited amount of request)

Almost all of the services will let you subscribe and send a certain amount of requests for a period of time (days) before you need to purchase a license. In our research, they are quite similar in functionality but data quality varies by a lot. Unfortunately, the variance is based on backends so some may work better than others for any given country. The only free service is OpenStreetMap (via Nominatim API), but if you send a vast amount of consecutive requests they will block your access.

You could deploy your own OpenStreetMap and install the Nominatim layer on top, but it does require a quite powerful server and a lot of time investment to get it up and running, as well as to maintain it – so be sure you really need it.

Reverse geocoding (technical example)

Let's use the free Nominatim API to make a reverse geocoding request from a hotel. Searching for hotels in 'Les Angles, Fr':

EANHotelID	Name	Address1	Address2	City	StateProvince	PostalCode	Country	Latitude	Longitude
388709	Lagrange Confort+ L'Orée Des Cimes	Avenue de Mont Louis		Les Angles		66210	FR	42.57622	2.06994
425093	Résidence Club mmv Les Angles Les Chalets de L'Isard	Route du Pla del Mir		Les Angles		66210	FR	42.56711	2.07032

If we use the first hotel, you will notice we do not know much about its address, as there is no StateProvince for it. Constructing a RESTful call we send in:

<http://nominatim.openstreetmap.org/reverse?format=json&lat=42.57622&lon=2.06994&zoom=18&addressdetails=1>

We got back a JSON response:

```
{
  "place_id": "9167865406",
  "licence": "Data © OpenStreetMap contributors, ODbL 1.0."
```

```

http://www.openstreetmap.org/copyright",
  "osm_type": "way",
  "osm_id": "88074349",
  "lat": "42.5751299",
  "lon": "2.0711922",
  "display_name": "Téléski Bamby, Rue du Tennis, Les Angles, Pirineos Orientales, Languedoc-Rosellón, Metropolitan France, 66210, France",
  "address": {
    "address29": "Téléski Bamby",
    "road": "Rue du Tennis",
    "village": "Les Angles",
    "county": "Pirineos Orientales",
    "state": "Languedoc-Rosellón",
    "country": "France",
    "postcode": "66210",
    "country_code": "fr"
  }
}

```

You can tell that the address is more detailed in this response (also disassembled in its elements) than the one we have on file. You could use this information as a seed to discover property matches in multi-supplier environments.

Multi-Hotel Supplier Lists

In the case of matching a list of hotels with the ID of another supplier, we have worked with two suppliers that sell these services. They are based on subscription lists; their listings include location information and we are mentioning it here as it been helpful in some partners implementations. Companies are:

- [GIATA](#) - GIATA Codes identify more than 362,000 individual hotels and hotel complexes, including details of the town/city and country.
- [Match2List](#) – They already have pre-created list of our inventory matched to other suppliers (you need to send them an email to request that product).

Conclusion

We have discussed the various geographical data points and tables that EAN makes available to our partner community. All the scripts you see here are available at our [github repository](#) repository under the /Geography sub directory of the scripts section.

Let us know if you find this whitepaper useful or if you need some more clarifications, or queries, please send us a note at: apihelp@expedia.com.

Til next time!

Jon C. Arce (jon.arce@gmail.com)