Mum, are we there yet?
Catchment Area Analysis for Retail

Albert Godfrind
Oracle
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

• The business problem
• The technology problem
• Geocoding
• Network Data Model
• Putting it all together
Application Scenario

• Hundreds of stores
• Millions of customers
• Run a mass mailing campaign
• Campaign targeted to customers of each store
• Identify the “Catchment Area” of each store:
  – The area from where customers can reach a store in less than 30 minutes.
The Problem ...

• Find all customers that are within a set distance from each store
• Need to compute N (stores) x M (customers) distances!
  – For 1000 stores and 10 million customers, that means 10 billion calculations!
• Straight line calculations are misleading
  – May be impossible (example: no bridge)
• Use road network
  – Use transportation time (not distance)
  – Driving or walking use different routes
More About the Problem ...

• Compute the routes on the actual road network using **drive times**
  – Requires a routable road network graph
  – May consider traffic statistics

• Need to compute \(N\) (stores) \(\times\) \(M\) (customers) **routes**
  – For 1000 stores and 10 million customers, that means **10 billion** route calculations!
  – Costly, even on powerful hardware

• You can pre-sort the data
  – Segment customers and stores using Voronoi diagrams
  – But that itself is a costly process
The Approach

1. Determine the geographic location of each store and customer from their addresses
   – This is a process called Geocoding.
   – Use Oracle Spatial’s Geocoder to determine the road segment for each address

2. Determine the Reverse Network Buffer for each store
   – Contains all the road segments from which a store can be reached within a set time
   – Use Oracle Spatial’s Network Data Model via its Java API

3. Correlate the resulting road segments using a regular join operation
   – Can be done in a single SQL statement, thanks to SQL Analytics functions.
Road Networks and Addresses in Oracle

• Reference data available from HERE (aka Navteq)
  – Available in “Oracle Data Format” as Transportable Tablespaces
  – Available for most countries worldwide

• This data set contains:
  – The road network, complete with geometries, connectivity and attribution
  – A dictionary of street names and attributes for geocoding
  – Geographic data for visualization: admin boundaries, water bodies, parks,, points-of-interest, ...

• Open data model
  – Each road segment includes attributes for length, travel time, accessibility
  – Fully documented and extensible, for example with your own attributes

• Also available from other suppliers (e.g. Tomtom)
Demo and Sample Data Available on OTN
Loading Customer Addresses from a CSV File

- The CSV Files are in directory “address_data_dir”
- Can process any number of files
- Defined as an “External Table”
- Load them using a “Create Table As Select” (CTAS) statement.
- Or use directly

```sql
CREATE TABLE customer_addresses (  
customer_id NUMBER,  
housenumber VARCHAR2(10),  
streetname VARCHAR2(50),  
city VARCHAR2(30),  
state VARCHAR2(20),  
zip VARCHAR2(10) )  
) ORGANIZATION EXTERNAL (  
TYPE ORACLE_LOADER  
DEFAULT DIRECTORY address_data_dir  
ACCESS PARAMETERS (  
    RECORDS DELIMITED BY NEWLINE  
    FIELDS TERMINATED BY ',',  
)  
LOCATION ( 'customers.csv' )  
);```
Geocoding

• Convert a street address into geographical coordinates
  – Input can be structured or unstructured

• Coordinates can be exact (“address points”) or computed by interpolation

• Also returns the road segment id (EDGE_ID) and relative location (PERCENT)

• Two available APIs
  – **PL/SQL**: best for batch processing – use it to geocode all your existing customers or stores. Can take advantage of parallelism.
  – **XML Web Service**: use it to geocode addresses provided by web applications

• A standard feature of Oracle Spatial and Graph Option
Example of a Geocoding Operation

```sql
SELECT SDO_GCDR.GEOCODE(
    'ODF_NA_Q312',
    SDO_KEYWORDARRAY('1500 Clay Street', 'San Francisco, CA'),
    'US', 'DEFAULT'
) FROM DUAL;
```

- **Schema that contains the address data**
- **Unstructured Address**
- **Country code**
- **Segment, position**
- **Geographic Coordinates**
- **Matchcodes**
Parallel Pipelined Table Functions

• Allows you to geocode millions of addresses by exploiting parallelism

• Input to the pipelined function is a **cursor**
  – Can pass any SELECT statement using a CURSOR expression

• Output is a **table**
  – Cast as a regular relational table using the TABLE() construct.
  – Can feed the result into a table

• Data flows continuously from input to output
Inside the Pipelined Function: Result Types

• Define an object type and array to hold the result of the function

```sql
CREATE OR REPLACE TYPE gc_result AS OBJECT (  
id NUMBER,  
geo_addr sdo_geo_addr
);
/

CREATE OR REPLACE TYPE gc_results AS TABLE OF gc_result;
/
```
Inside the Pipelined Function

```
CREATE OR REPLACE function geocode_pipe (source_table_cursor IN SYS_REFCURSOR)
RETURN gc_results
  DETERMINISTIC
  PIPELINED
  PARALLEL_ENABLE (PARTITION source_table_cursor BY ANY)
IS
...
```

- The function is declared as PIPELINED and PARALLEL_ENABLE
- Its input is a cursor, its output is a table
Inside the Pipelined Function

```sql
IS
  TYPE number_table IS TABLE OF NUMBER;
  TYPE string_table IS TABLE OF VARCHAR2(256);

  id_table    number_table;
  line_1_table string_table;
  line_2_table string_table;

  gc gc_result := gc_result(0,null);
```

Inside the Pipelined Function

```
BEGIN
    LOOP
        FETCH source_table_cursor
            BULK COLLECT INTO id_table, line_1_table, line_2_table
            LIMIT 1000;
        EXIT WHEN id_table.count = 0;
        FOR i IN id_table.. id_table LOOP
            gc.id := id_table (i);
            gc.geo_addr := SDO_GCDR.GEOCODE ('ODF_NA_Q312',
                sdo_keywordarray(line_1_table(i), line_2_table(i)), 'US', 'default');
            PIPE ROW (gc);
        END LOOP;
    END LOOP;
END;
```

- Fetch the input addresses in batches of 1000
- Geocode each address
- Send the result to the caller
Using the Pipelined Function

```sql
CREATE TABLE customers_gc as
SELECT /*+ parallel (16) */
  g.id customer_id, g.geo_addr.longitude, g.geo_addr.latitude,
  g.geo_addr.edgeid link_id, g.geo_addr.percent percentage
FROM TABLE(geocode_pipe(
   CURSOR(SELECT customer_id, housenumber ||' '|| streetname,
           city ||' '|| state ||' '|| zip
           FROM customer_addresses)
   ) g;
```

- Scales perfectly on Exadata, but also on conventional hardware
- Geocoding US addresses on Exadata X4-2 Half-Rack: over **23,000 addresses/second**
- Geocode 1 million customer addresses in less than a minute
Geocoding Result: Table CUSTOMERS_GC

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>LONGITUDE</td>
<td>NUMBER</td>
</tr>
<tr>
<td>LATITUDE</td>
<td>NUMBER</td>
</tr>
<tr>
<td>LINK_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>PERCENTAGE</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

- CUSTOMER_ID is the customer identifier
- LINK_ID is the identifier of the network link (road segment) on which the customer is located
- PERCENTAGE is the relative location on that link (0 to 1).
Network Data Model

• Data model for managing and analyzing network graphs
  – Roads, transportation, utilities, ...

• Formed with **nodes** and **links**
  – Directed or un-directed
  – With or without geometries
  – Any user attributes (costs, access restrictions, ...)

---

Copyright © 2014 Oracle and/or its affiliates. All rights reserved.
Network Data Model: Analysis

- Operates on a memory-resident copy of the network
  - Network subsets (partitions) loaded “on demand”
- Network analysis functions accessed via a Java API
Network Data Model: Analysis Functions

- Shortest path analysis
- Nearest neighbor analysis
- Within cost analysis
- Network Buffer (forward and reverse)
- Traveling salesman problem
- Reachable/Reaching nodes
- K-shortest paths analysis
- ...

Copyright © 2014 Oracle and/or its affiliates. All rights reserved.
Using the NDM Java API for Solving our Problem

✓ Prepare and setup
  – Create a **NetworkAnalyst** object

✓ Define a custom cost function
  – compute link travel time from link length and speed limit

✓ Determine the catchment area for each store (using **reachingNetworkBuffer**)
  – Finds all road segments in the catchment area

✓ Store the results in a table for further analysis
  – All road segments identifiers with drive time for each catchment area
private static NetworkIO networkIO;
private static NetworkAnalyst analyst;

// Open database connection
conn = LODNetworkManager.getConnection(dbUrl, dbUser, dbPassword);

// Get network input/output object
networkIO = LODNetworkManager.getCachedNetworkIO(
    conn, networkName, networkName, null);

// Get network analyst
analyst = LODNetworkManager.getNetworkAnalyst(networkIO);
Drive Time Cost Function

- Each link has a “speed limit” attribute (in m/sec)
- Function computes link drive time from link length and speed limit

```java
public class LinkTravelTimeCalculator implements LinkCostCalculator {
    int[] defaultUserDataCategories = {UserDataMetadata.DEFAULT_USER_DATA_CATEGORY};
    public LinkTravelTimeCalculator () {}
    public double getLinkCost(LODAnalysisInfo analysisInfo) {
        LogicalLink link = analysisInfo.getNextLink();
        double speed = ((Double)link.getUserData(0).get
                        (RouterPartitionBlobTranslator11gR2.USER_DATA_INDEX_SPEED_LIMIT)).doubleValue();
        return (link.getCost()/speed);
    }
}
```

- Tell the network analyst to use the cost function

```java
LinkCostCalculator[] lccs = {new LinkTravelTimeCalculator};
analyst.setLinkCostCalculators(lccs);
```
Reverse Network Buffer

• Compute a reverse network buffer for each store

```java
PointOnNet startPoint = new PointOnNet(storeLinkId, storePercentage);
PointOnNet[] startPoints = {startPoint};
NetworkBuffer buffer = analyst.reachingNetworkBuffer(startPoints, cost, null);
```

• Save the buffer

```java
tableNamePrefix = "STORE_BUFFER_";
networkIO.writeNetworkBuffer(buffer, storeId, tableNamePrefix);
```

• Buffers are now in table STORE_BUFFER_NBL$
Network Buffer Result: Table **STORE_BUFFER_NBL$**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFER_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>LINK_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>START_PERCENTAGE</td>
<td>NUMBER</td>
</tr>
<tr>
<td>END_PERCENTAGE</td>
<td>NUMBER</td>
</tr>
<tr>
<td>START_COST</td>
<td>NUMBER</td>
</tr>
<tr>
<td>END_COST</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

- BUFFER_ID is the store identifier
- Percentages are between 0 and 1
- Start % usually 0, end % usually 1
  - except for boundary links
- Cost from start node to store.
- Cost from end node to store.
CREATE TABLE store_customer_distances AS
WITH

step1 AS (  
SELECT c.customer_id AS c_customer_id,  
c.link_id AS c_link_id,  
c.percentage AS c_percentage,  
b.buffer_id AS b_buffer_id,  
b.link_id AS b_link_id,  
b.start_percentage AS b_start_percentage,  
b.end_percentage AS b_end_percentage,  
b.start_cost AS b_start_cost,  
b.end_cost AS b_end_cost  
FROM customers c, store_buffer_nbl$ b  
WHERE c.link_id = b.link_id  
)}
Putting it all Together (2)

```sql
step2 AS ( 
    SELECT c_customer_id, b_buffer_id,  
        b_start_cost + (c_percentage - b_start_percentage) * (b_end_cost - b_start_cost) / (b_end_percentage - b_start_percentage) cost  
    FROM step1  
    WHERE c_percentage BETWEEN b_end_percentage AND b_start_percentage  
),

step3 AS (  
    SELECT c_customer_id, b_buffer_id, cost,  
        row_number() over (partition by c_customer_id order by cost) rn  
    FROM step2  
)

SELECT /*+ parallel 16 */ customer_id, buffer_id as store_id, cost/60 as minutes  
FROM step3  
WHERE rn = 1;
```

---

**Step 2**  
Compute cost for each customer

**Step 3**  
Choose lowest cost

**Final**  
Save results
The Result: Table **STORE_CUSTOMER_DISTANCES**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>STORE_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>MINUTES</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

- One row per customer
- Identifier of nearest store for that customer
- Travel time from that customer to the store, in minutes

Ready for further analyzes
Visualizing the Network (MapViewer)

Simple Network View

View in Oracle Maps
Conclusions

• The example shows that you can perform complex analysis efficiently
  – Correlate hundreds of stores and millions of customers

• The technologies used can be applied to many cases
  – Great potential for spatial analyses, data warehousing, ...

• Try it yourself using a ready-for-use virtual machine (here)
  – Database, sample data, scripts, java code, ...

}[415x94]Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |
The Spatial and Graph SIG

• The SIG promotes interaction and communication that can drive the market for spatial technology and data

• Members connect and exchange knowledge via online communities and at annual conferences and events

• Meet us at the Summit
  • Morning Receptions
    • Tuesday and Wednesday / 7:45 to 8:30 a.m. / Registration Area
  • Birds of a Feather Session
    • Tuesday / 12 to 1 p.m. / Lunch Room

• Join us online
  • LinkedIn (search for “LinkedIn Oracle Spatial”)
  • Google+ (search for “Google+ Oracle Spatial”)
  • IOUG SIG (sign up for free membership through www.ioug.org)
  • OTN Spatial – Communities (search for “Oracle Spatial and Graph Community”)

• Contact the Board at oraclespatialsig@gmail.com
Resources

• Oracle Technology Network:  
  www.oracle.com/technetwork/database/options/spatialandgraph  

• blogs.oracle.com ➔ oraclespatial ➔ oracle_maps_blog ➔ bigdataspatialgraph
Resources on Big Data Spatial and Graph

- Oracle Big Data Spatial and Graph on Oracle.com: https://www.oracle.com/database/big-data-spatial-and-graph


- Blog (technical examples and tips): https://blogs.oracle.com/bigdatabigsquidgraph

Hardware and Software
Engineered to Work Together