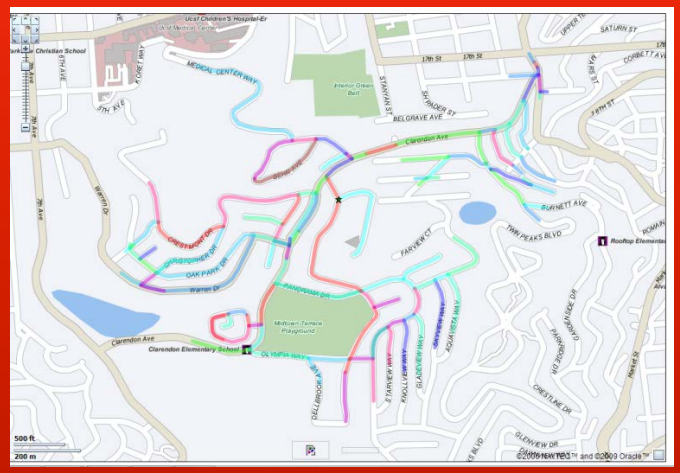
ORACLE

May 21, 2014
Walter E. Washington Convention Center
Washington, DC USA

How To Build a Drive Time Analysis Application - Workshop

Daniel Geringer
Senior Software Development Manager



Program Agenda

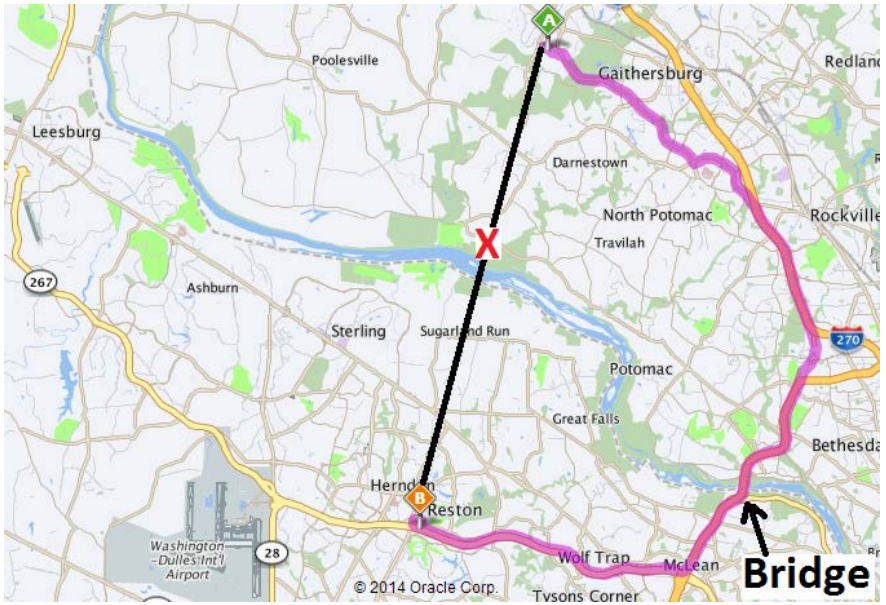
- Today's Goal – Overview of drive time requirement
- Overview of VM for workshop
- HERE Sample for San Francisco
- Parallel Geocoding
- Oracle Spatial & Graph - Network Graph Overview
- Sample Java Application – Overview / Compile / Run

Today's Goal:

Learn How to Build a Scalable Network Analysis Application

Why Drive Time Analysis Is Important

- Reachabiltiy given a constraint (time or distance) **should consider a road network**.
- "As the crow flies" computations can be misleading.
- For example, they may cross rivers where there are no bridges.



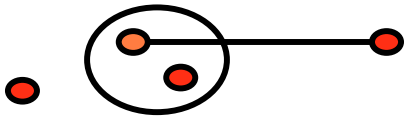
 | |

Spatial Analysis Versus Network Analysis

- Oracle Locator and Oracle Spatial can solve spatial proximity problems, sometimes referred to “as the crow flies” analysis.
- Another type of analysis that is required by users and applications is network analysis.
- Network applications deal with the connectivity of features.

Spatial Closest feature

(based on distance)



Verses

Network Closest feature

(based on connectivity and cost)

Marketing Requirement
Road Network, Stores, and Customers

- Store locations in red
- Street network for the US in black
- E-mail fliers to millions of customers from their closest store.

More Than One Way To Solve

But not all approaches are optimal

- As the crow flies computations often assign wrong store to a customer.
- Given 1,000,000 customers, and 100 stores, compute drive time every combination? Takes a very long time.
- Voronoi Diagrams to group customers and stores and reduce drive time computations.
- Preprocessing may take days.

ORACLE

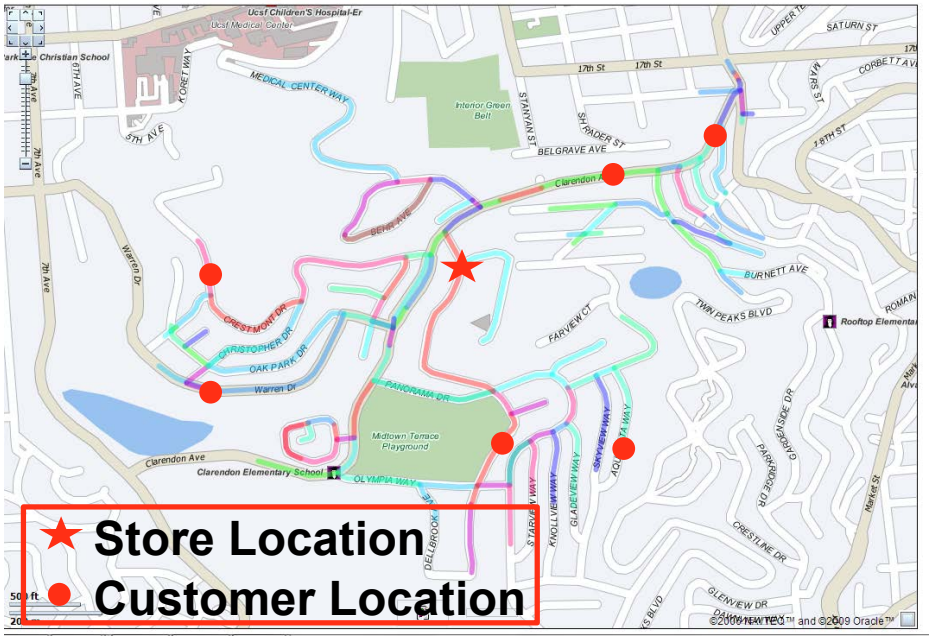
THINK OUT OF THE BOX

Large Scale Drive Time/Distance Analysis

Oracle Strategy

For millions of customers, find closest store within a specified drive time

- Same underlying data for geocoder and road network
- Customers geocode as link id and percentage (instead of longitude/latitude)
- 20 minute “reverse“ Network Buffer from each store generates all possible paths
- Each persisted path includes:
  - Covered link IDs, nodes ID, and associated costs
- Single database query to find closest store and drive time/distance for each customer (join on link_id)



Database Information for VM

Start Database and Listener

- Password for VirtualBox oracledemo user is oracledemo
- Database should already be running. If not, from a terminal window, run:
  - sqlplus / as sysdba
  - SQL> startup
  - *Wait for message: Database opened.*
  - SQL> exit
- Listener should already be running. From a terminal window, run:
  - lsnrclt status (to check if listener is running)
  - lsnrctl start (to start listener)
- Both SQL*Plus and SQL Developer are installed in the VM

VM Linux Users

Linux User	Password
oracle	oracledemo
root	oracledemo

Database Users

- Primarily you will be working with the sys, oracledemo and here_sf database accounts.

Database User	Password	Description
sys	**oracledemo**	**sys admin user**
system	oracledemo	system admin user
oracledemo	**oracledemo**	**You will work in this account**
HERE_SF	**HERE_SF**	**Owner of HERE San Francisco sample data**
mvdemo2	mvdemo2	MapViewer demo data owner
storm	storm	MapViewer storm demo data owner
ndmdemo	ndmdemo	Network Graph demo user

Aliases created for you

- Just typing the database username will connect through SQL*Plus
  - alias **oracledemo**='sqlplus oracledemo/oracledemo'
  - alias **sys**='sqlplus sys/oracledemo as sysdba'
  - alias **system**='sqlplus system/oracledemo‘
  - alias **HERE_SF**='sqlplus HERE_SF/HERE_SF'

HERE
San Francisco Sample Data

HERE – San Francisco Sample Data Set

Worldwide coverage available from HERE

- Pre-downloaded and installed on the VM. Very easy to install.
- Downloadable from the Oracle Technology Network (must accept license)
  - http://www.oracle.com/technetwork/database/options/spatialandgraph/downloads
  - Spatial Features Partners' Data Downloads
  - HERE Map Content Sample in Oracle Delivery Format for San Francisco
- Contains Geocoding, Routing and Mapping data sets.
- **Geocoding road_segment_id** matches **Router edge_id**
  - Geocoder GC_ROAD_SEGMENT_NVT (ROAD_SEGMENT_ID) are always positive numbers
  - Router EDGE(EDGE_ID) can be positive or negative to denote direction.

HERE – San Francisco Sample Data Set

- San Francisco sample contains:
  - Geocoding data (street centerline and roof top)
  - Routing data, trucking data, traffic patterns, multi-modal
  - Mapping data
- Worldwide geographies also available

Parallel Enabled Geocoding

Oracle Spatial & Graph - Geocoder

- Geocoder is included in your Oracle Spatial and Graph license.
- Open data model for Geocoder reference data
- If you have reference data, you can populate the data model yourself
- If you don't have the reference data, Oracle Partners sell it in Transportable Tablespace format (plug and play data).
  - HERE
  - Tom Tom
  - ADCI
  - others

Oracle Spatial and Graph Geocoder

- Forward / Reverse / Street Centerline / Rooftop (point based) support
- In database geocoding –
  - PL/SQL APIs
  - Optimal for parallel enabled batch geocoding
  - For batch processing, leverage parallel enabled pipeline table functions
- Web service based geocoding
  - Java servlet based with XML geocoding APIs
  - Deployed in J2EE container
  - Optimal for non-batch request in web based applications.
  - Can perform batch processing too

In Database sdo_gcdr.Geocode API

– Accepts an **unparsed address** as an array of strings

```
SELECT sdo_gcdr.Geocode ('HERE_SF',
                         sdo_keywordarray( '33 New Montgomery St.' ,
                                           'San Francisco , CA 94105' ),
                         'US','DEFAULT')
FROM dual;
```

In Database sdo_gcdr.Geocode_Addr API

– Accepts a **parsed address** as input

-- Higher match rate if you perform the parse and use this API

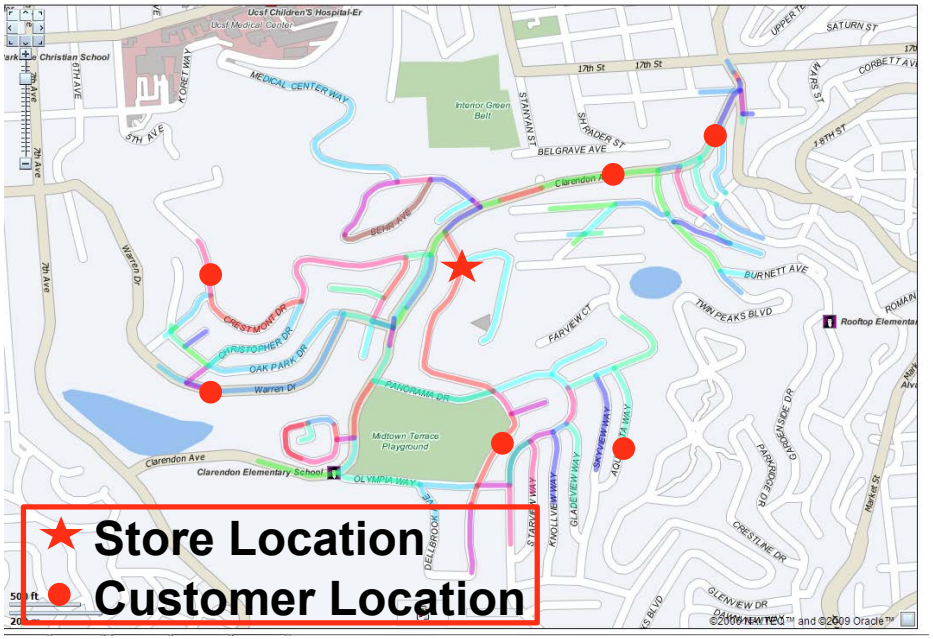
```
SELECT sdo_gcdr.Geocode_Addr ('HERE_SF',
                              SDO_GEO_ADDR(0, null, null,
                                           'New Montgomery St.' , NULL, NULL,
                                           'San Francisco' , NULL,
                                           'CA' , 'US' ,
                                           '94105' ,  NULL, NULL, NULL,
                                           '33' ,  NULL, NULL, NULL, NULL,
                                           NULL, NULL, NULL, NULL, NULL,
                                           NULL, NULL,
                                           'DEFAULT'  ,NULL,NULL))
FROM dual;
```

ORACLE

Large Scale Drive Time/Distance Analysis
Strategy

For millions of customers, find closest store within a specified drive time

- Same underlying data for geocoder and road network
- Customers geocode as link id and percentage (instead of longitude/latitude)
- 20 minute “reverse“ Network Buffer from each store generates all possible paths
- Each persisted path includes:
  - Covered link IDs, nodes ID, and associated costs
- Single database query to find closest store and drive time/distance for each customer (join on link_id)



 | |

Geocode Result

- Result returns both:
  - Percent (between 0 and 1) and Edge ID. In this example (.12, 23612131)
  - Longitude, Latitude
- We want Percent, Edge ID

SDO_GEO_ADDR(0, SDO_KEYWORDARRAY(), NULL,
'New Montgomery St', NULL, NULL, 'SAN FRANCISCO',
'SAN FRANCISCO', 'CA', 'US', '94105', NULL, '94105', NULL,
'33', 'NEW MONTGOMERY', 'ST', 'F', 'F', NULL, NULL, 'R',
.12, 23612131, '??X?#ENUT?B281CP?', 1, 'DEFAULT',
-122.40158, 37.78835, '??010101010??000?', 8307)

Often Addresses Are In CSV File Format

- Oracle External Tables can point to a CSV file
- External tables are read only
- Setup is similar to a SQL*Loader control file

External Table For Address CSV Files

- CSV file resides in address_data_dir directory
- Multiple CSV files can be listed
- Select from the external table like any other table

```
CREATE TABLE customer_addresses_ext
   (in_customer_id      NUMBER,
    in_housenumber      VARCHAR2(1000),
    in_streetname       VARCHAR2(1000),
    in_city             VARCHAR2(100),
    in_state            VARCHAR2(100),
    in_zip              VARCHAR2(100))
  ORGANIZATION EXTERNAL
   (TYPE ORACLE_LOADER
    DEFAULT DIRECTORY address_data_dir
    ACCESS PARAMETERS
      (RECORDS DELIMITED BY NEWLINE
       NOLOGFILE
       FIELDS TERMINATED BY '|')
    LOCATION (
     -- This can be a comma delimted list of csv files
     'customers.csv'));
```

ORACLE

Parallel Enabled Pipeline Table Function

Excellent for Batch Processing

- Parallelize a function that's called a massive amount of times.
  - Batch geocoding (sdo_gcdr.geocode_addr)
  - Batch reverse geocoding (sdo_gcdr.reverse_geocode)
- Pipeline Table Function returns a table of results
  - Table of geocodes
  - Table of reverse geocodes

Parallel Enabled Pipeline Table Function
How does it work?

- Define cursor that selects the input to the batch process:
  - SELECT address
    FROM customer_addresses
    WHERE state = 'NY';
  - SELECT vehicle_location
    FROM vehicles
    WHERE sdo_anyinteract (location,:region) = 'TRUE'
- Parallel Query distributes the batch process over a specified number of database cores.

Parallel Batch Geocoding Example


```
ALTER SESSION ENABLE PARALLEL QUERY;
ALTER SESSION ENABLE PARALLEL DDL;

-- Additional attributes available like corrected a.house number, a.streetname, etc
--   are commented out.  Uncomment if you would like them returned too.
DROP TABLE customers;
CREATE TABLE customers NOLOGGING AS
SELECT /*+ parallel (16) */
      a.id customer_id, a.longitude, a.latitude,
      --a.housenumber,
      --a.streetname,
      --a.settlement,
      --a.region,
      --a.postalcode,
      --a.matchcode,
      a.edgeid link_id, a.percent percentage
FROM TABLE( geocode_utils.geocode_parsed (CURSOR( SELECT in_customer_id,
                                                         in_housenumber,
                                                         in_streetname,
                                                         in_city,
                                                         in_state,
                                                         in_zip
                                                  FROM customer_addresses_ext ),
                                          'HERE_SF')) a;
```

Parallel Pipelined Table Function

Input cursor

Geocoding Lab

Step 1 – Grant SELECT on Geocoder Tables

- HERE San Francisco sample was pre-populated into HERE_SF user.
- You will be working under the oracledemo database user
- Log in as HERE_SF, and grant SELECT on all tables and view to oracledemo
  - cd /home/oracle/WORKSHOP/GEOCODE
  - grant.sh
- Now you can connect as oracledemo to geocode instead of connecting as HERE_SF

Step 2 – Test a Geocode

- From the /home/oracle/WORKSHOP/GEOCODE directory:
  - Log in as oracledemo and run a test geocode
  - oracledemo
  - SQL> @test_one_geocode.sql
- First geocode in a session initializes the geocoding stored procedure, and takes a little longer to run.
- Subsequent geocodes will be very fast. Run it again.
  - SQL> @test_one_geocode.sql

Step 3 – Load geocode_utils

- geocode_utils is a package that contains parallel pipelined table functions for bulk geocoding.
- To install geocode_utils, as the oracledemo user:
  - SQL> @geocode_utils_setup.sql

Step 4 – Geocode Customers

- This VM is set up as having 2 processors. Increase VM processors if your host machine has more processors.
- Increase parallel degree in geocode_customers.sql if you have more than 2 processors
- To observe parallel threads, from a terminal, run top
  - top
- As oracledemo, geocode 77,216 customers. Results will be in a table called customers.
  - SQL> @geocode_customers.sql
- Notice processes in top (ora_p000 and ora_p001) consume most of the CPU. On a non-VM or VM with adequate resources, parallel processes will ramp up much faster.

Geocode Times On Exadata X4-2 1/2 RAC

- X4-2 with 96 cores
- Geocoded 77216 addresses in 3.32 seconds
- 23,257 geocodes per second
- Works on commodity hardware too

Step 5 – Geocode Stores

- Increase parallel degree in geocode_customers.sql if you have more than 2 processors
- As oracledemo, run geocode_stores.sql to geocode 12 stores. Result will be in a table called stores.
  - SQL> @geocode_stores.sql

Oracle Spatial and Graph Network Graph Overview

What Is Oracle Spatial & Graph Network Graph?

- An open data model to store and analyze network data.
- Connectivity is determined using nodes and links:
  - Each link has a start node and an end node.
  - Links and/or nodes can have costs
  - Links can be one way or bi-directed

What Is Oracle Spatial & Graph Network Graph?

- Analysis is based on connectivity and optionally cost information.
- Network analyses includes:
  - Shortest path analysis
  - Nearest neighbor analysis
  - Within cost analysis
  - Network Buffer (forward and reverse)
  - Traveling salesman problem
  - Reachable/Reaching nodes
  - K-shortest paths analysis

VERY SIMPLE DATA MODEL

The Node Table (or View)

- Optional columns marked in red.
- Additional user data (custom data) columns can be added as needed.

```
SQL> desc MYNETWORK_NODE$;
 Name                          Type
 ----------------------------- ------------
 NODE_ID                       NUMBER (Primary Key)
 COST                          NUMBER
 ACTIVE                        VARCHAR2(1)
 GEOMETRY                      SDO_GEOMETRY
```

The Link Table (or View)

- Optional columns marked in red
- Additional user data columns can be added
- Links can be one way or bi-directed
- If bi-directed with a different cost in either direction, add another link.

```
SQL> desc MYNETWORK_LINK$;

 Name                  Type
 --------------------  -------------
 LINK_ID               NUMBER (Primary Key)
 START_NODE_ID         NUMBER
 END_NODE_ID           NUMBER
 ACTIVE                VARCHAR2(1)
 LINK_LEVEL            NUMBER
 COST                  NUMBER
 GEOMETRY              SDO_GEOMETRY
```

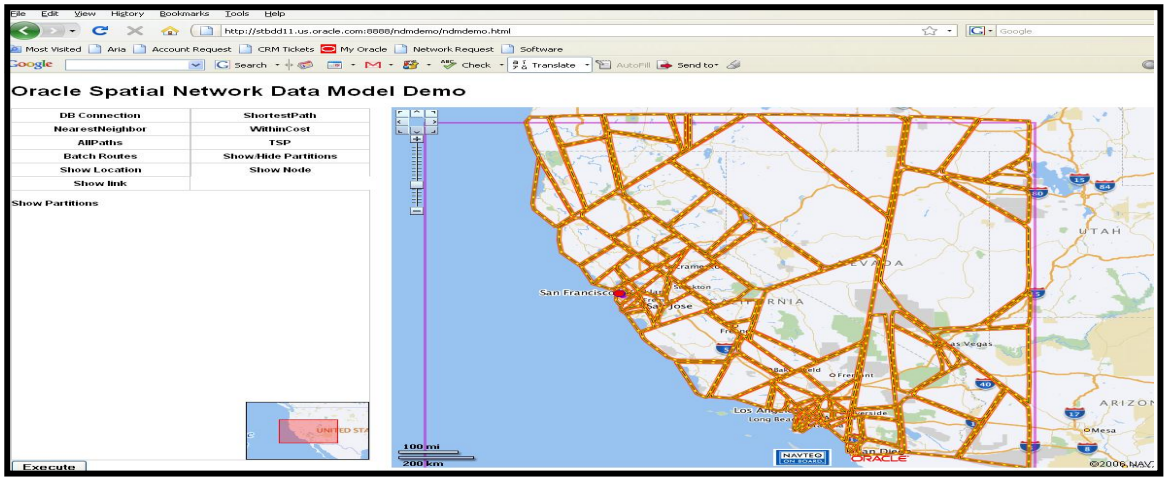
Network Partition Table for Very Large Networks Load on Demand (Optional)


```
SQL> desc MYNETWORK_PART$;
 Name                          Type
 ----------------------------- ------------
 NODE_ID                       NUMBER
 LINK_LEVEL                    NUMBER
 PARTITION_ID                  NUMBER
```

- If no network partition table, entire network is loaded into memory as a single partition.
- One row for each node.
- Nodes repeated for each hierarchy they belong to (`LINK_LEVEL`).
- You can manually populate a network partition table.
- `SDO_NET.SPATIAL_PARTITION`, partitions a spatial network for you (creates the partition table).

Visualizing Partition Boundaries

- Observe smaller partition boundaries in denser portions of the network (urban areas).
- The number of nodes in each partition is still balanced.



Partition BLOB Table (Optional)


```
SQL> desc MYNETWORK_PBLOB$;
 Name                                      Type
 ----------------------------------------- ------------
 LINK_LEVEL                                NUMBER
 PARTITION_ID                              NUMBER
 BLOB                                      BLOB
 NUM_INODES                                NUMBER
 NUM_ENODES                                NUMBER
 NUM_ILINKS                                NUMBER
 NUM_ELINKS                                NUMBER
 NUM_INLINKS                               NUMBER
 NUM_OUTLINKS                              NUMBER
 USER_DATA_INCLUDED                        VARCHAR2(1)
```

- Partition `BLOB`s store network partitions in a binary format.
- The binary format is much faster to load into memory.

Network Metadata
Enter One Row Per Network

- HERE San Francisco sample enters one row in setup.txt.
- Network name is HERE_SF_NET
- Includes these views too:
  - HERE_SF_NET_NODE$
  - HERE_SF_NET_LINK$
  - HERE_SF_NET_PART$
  - HERE_SF_NET_PBLOB$

```
SQL> desc user_sdo_network_metadata;
 Name                                      Type
 ----------------------------------------- ----------------
 NETWORK                                   VARCHAR2(24)
 NETWORK_ID                                NUMBER
 NETWORK_CATEGORY                          VARCHAR2(12)
 GEOMETRY_TYPE                             VARCHAR2(24)
 NETWORK_TYPE                              VARCHAR2(24)
 NO_OF_PARTITIONS                          NUMBER
 NODE_TABLE_NAME                           VARCHAR2(32)
 NODE_GEOM_COLUMN                          VARCHAR2(32)
 NODE_COST_COLUMN                          VARCHAR2(32)
 LINK_TABLE_NAME                           VARCHAR2(32)
 LINK_GEOM_COLUMN                          VARCHAR2(32)
 LINK_DIRECTION                            VARCHAR2(12)
 LINK_COST_COLUMN                          VARCHAR2(32)
 PATH_TABLE_NAME                           VARCHAR2(32)
 PATH_GEOM_COLUMN                          VARCHAR2(32)
 PATH_LINK_TABLE_NAME                      VARCHAR2(32)
 PARTITION_TABLE_NAME                      VARCHAR2(32)
 PARTITION_BLOB_TABLE_NAME                 VARCHAR2(32)
```

The Java Application Example

Get Connection and Set Logging Level


```
// opening connection
conn = LODNetworkManager.getConnection(dbUrl, dbUser, dbPassword);

// Possible values: FATAL, ERROR, WARN, INFO, DEBUG, FINEST
// For debugging, set to FINEST
setLogLevel(logLevel);
```

Read Configuration File

- Slightly different format for configuration file in 11g and 12c
- This VM uses a 12c formatted configuration file
- File is located in classes/nbuffer/LODConfigs.xml since program is compiled with "–d classes" directive

```
String configXmlFile = "nbuffer/LODConfigs.xml";

// load user specified LOD configuration (optional),
// otherwise default configuration will be used
InputStream config = ClassLoader.getSystemResourceAsStream(configXmlFile);
LODNetworkManager.getConfigManager().loadConfig(config);
```

Configuration File

- For HERE data, specify RouterPartitionBlobTranslator11gR2
- Legacy partition blob format used by data providers (before Network Graph)
- If you create your own network, use PartitionBlobTranslator11gR2

```
<LODConfig globalNetworkName="HERE_SF_NET" networkName="HERE_SF_NET">
  <networkIO>
    <batchSize>10000</batchSize>
    <geometryTolerance>0.05</geometryTolerance>
    <readPartitionFromBlob>true</readPartitionFromBlob>
    <partitionBlobTranslator>
      <className> oracle.spatial.router.ndm.RouterPartitionBlobTranslator11gR2 </className>
      <parameters></parameters>
    </partitionBlobTranslator>
```

Configuration File (continued)

- userData is custom data associated with network data (nodes and links)
- For example speed limit
- userData is categorized
- User Data associated with category 0 is stored in partition blobs.
- HERE San Francisco sample stores speed limit userData in category 0

```
<userDataIO categoryId="0">
  <className>oracle.spatial.network.lod.LODUserDataIOSDO</className>
    <parameters>
    </parameters>
</userDataIO>
```

Configuration File (continued)

- Earlier we discussed network partition generation for very large networks
- Nodes can be associated with a link level. For example, level 1 are detailed roads, level 2 are highways, etc…
- For large networks, if you have enough memory, increase maxNodes

```
<cachingPolicy linkLevel="1">
    <maxNodes>200000</maxNodes>
    <residentPartitions></residentPartitions>
    <flushRule>
      <className>oracle.spatial.network.lod.LRUCachingHandler</className>
      <parameters></parameters>
    </flushRule>
</cachingPolicy>
</networkIO>
```

Initialize Analyst

- NetworkAnalyst class contains network analysis methods
- Create an analyst instance

```
private static NetworkIO networkIO;
private static NetworkAnalyst analyst;

//get network input/output object
networkIO = LODNetworkManager.getCachedNetworkIO(
                                conn, networkName, networkName, null);
//get network analyst
analyst = LODNetworkManager.getNetworkAnalyst(networkIO);
```

Implement Time Based Link Cost Calculator

Default Calculator Is Distance Based


```
public class LinkTravelTimeCalculator   implements LinkCostCalculator {
    int [] defaultUserDataCategories = {UserDataMetadata.DEFAULT_USER_DATA_CATEGORY};
    public LinkTravelTimeCalculator () {}

    public double getLinkCost(LODAnalysisInfo analysisInfo)  {
       LogicalLink link = analysisInfo.getNextLink();
      // speed in meters/second
      double speed =
         ((Double)link.getUserData(0).get
         (RouterPartitionBlobTranslator11gR2.USER_DATA_INDEX_SPEED_LIMIT)).doubleValue();
     return (link.getCost()/speed);      // distance/speed is travel time in seconds  }
```

Set Link Cost Calculator For Analyst

- Default link cost calculator is distance based
- Change it to be travel time based

```
// Save old link cost calculator to reset it later
LinkCostCalculator[] oldlccs = analyst.getLinkCostCalculators();

// Set new travel time based link cost calculator
LinkCostCalculator[]  lccs = {new LinkTravelTimeCalculator() };
analyst.setLinkCostCalculators(lccs);
```

Defined on previous slide

Get Each Store link_id From Stores Table

Account For Negative link_id


```
WITH
  part1 AS ( SELECT a.store_id, b.link_id,  a.percentage
              FROM stores a,
                   here_sf.here_sf_net_link$ b
             WHERE a.link_id = b.link_id
                UNION ALL
             SELECT a.store_id, b.link_id, 1 - a.percentage
             FROM stores a,
                  here_sf.here_sf_net_link$ b
             WHERE a.link_id = b.link_id),
  part2 AS (SELECT store_id,
                   link_id,
                   percentage,
                   row_number() OVER (PARTITION BY store_id ORDER BY store_id, link_id DESC) r_n
            FROM part1)
SELECT store_id, link_id, percentage
FROM part2
WHERE r_n = 1;
```

- Geocoder always positive link_id.
- here_sf_net_link$ match can be positive, negative or both
- Sign determines direction
- If negative, (1 – percentage)

- SQL Analytics to group and order by store_id
- Only keep one match

Compute Reverse Network Buffer

- Previous slide query returns store_id's with respective link_id and percentage
- For each store, compute a reverse network buffer
- For LinkTravelTimeCalculator, cost is specified in seconds

```
// Generate reverse network buffer, cost specified in seconds
PointOnNet startPoint = new PointOnNet(startLinkId, percentage);
PointOnNet[] startPoints = {startPoint};

NetworkBuffer buffer = analyst.reachingNetworkBuffer (startPoints, cost, null);
```

Write Reverse Network Buffer Results To A Table

- networkIO.writeNetworkBuffer creates the following tables if they don't exist
  - SF_NBCL$ - Contains
  - SF_NBCN$ - Contains
  - SF_NBL$ - Contains network buffer links (buffer_id is store_id)
  - SF_NBN$ - Contains network_buffer nodes
  - SF_NBR$ - Contains network buffer radius
- Pre-create _NBL$ and _NBN$ with no indexes for faster inserts/deletes

```
tableNamePrefix = "SF_";
networkIO.writeNetworkBuffer(buffer, (long)storeId, tableNamePrefix);
```

Link Information In SF_NBL$

- BUFFER_ID is store_id
- Each link contains:
  - Percentage is a value between 0 and 1
  - Start % usually 0, unless customer and store are on the same link.
  - End % usually 1, except for boundary links
  - Cost from start node to store.
  - Cost from end node to store.

```
SQL> describe  sf_nbl$
 Name                          Type
 ----------------------------- --------
 BUFFER_ID                     NUMBER
 LINK_ID                     NUMBER
 START_PERCENTAGE              NUMBER
 END_PERCENTAGE                NUMBER
 START_COST                    NUMBER
 END_COST                      NUMBER
```

Compile and Run the Java Application Lab

Compile Java Application

- Source code in:
  - PersistentNetworkBuffer.java
  - LinkTravelTimeCalculator.java
- PersistentNetworkBuffer.java declared inside a package:
  - package nbuffer;
- Compile with “javac -d classes” parameter
- Classes are generated in classes/nbuffer directory
  - cd /home/oracle/WORKSHOP/SOURCE_CODE_EXAMPLE
  - compile.sh

Generate Reverse Network Buffers

- To run application:
  - cd /home/oracle/WORKSHOP/SOURCE_CODE_EXAMPLE
  - run.sh
- run.sh executes the following command

```
java -cp $CLASSPATH nbuffer.PersistentNetworkBuffer \
-dbUrl "jdbc:oracle:thin:@localhost:1521/ora12c.oracledemo.com" \
-dbUser oracledemo \
-dbPassword oracledemo \
-networkName HERE_SF_NET \
-networkOwner HERE_SF \
-cost 1200 \
-tableNamePrefix SF \
-readFromTable true \
-inputTable STORES \
-logLevel ERROR
```

Visualize Network Buffers

Start WebLogic Server (WLS)

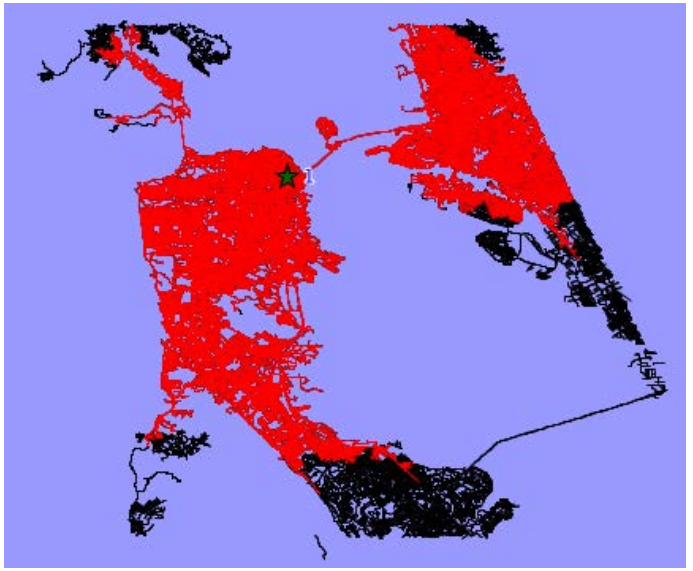
- Start WLS and pre-deployed servlets:
  - MapViewer
  - MVDemo – MapViewer demo
  - Geocoder – Web based geocoder service used by NDM Tutorial
  - NDM Tutorial – Network Graph tutorial
  - cd /home/oracle/Mapviewer
  - See README.txt to start WLS and pre-deployed servlets

Visualize Store Buffers

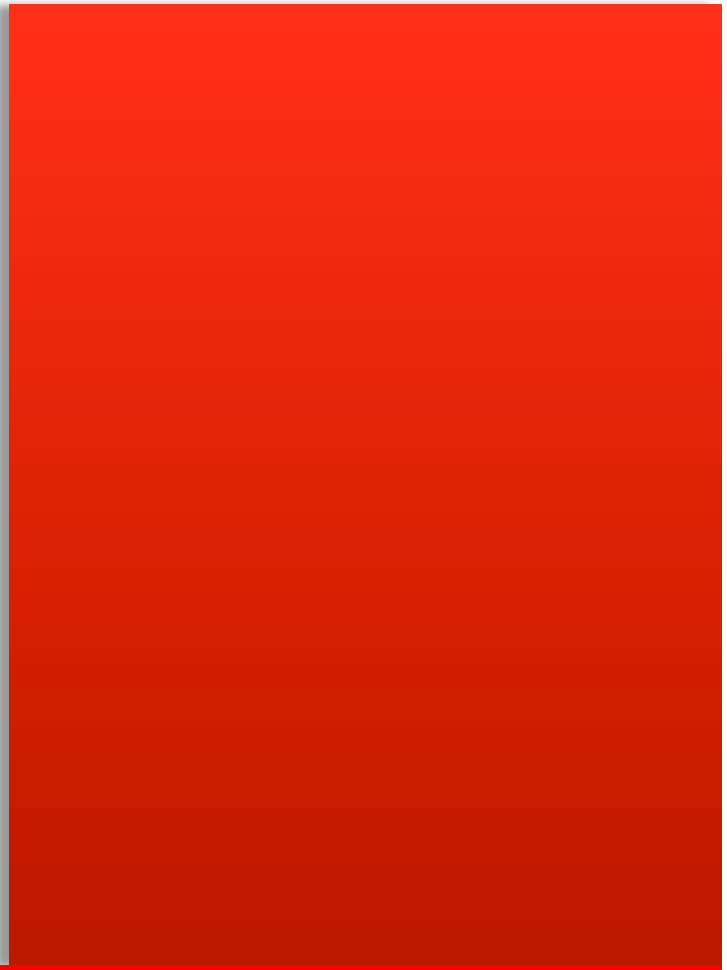
- To see the links covered by each store:
  - Start firefox
  - Under Bookmarks, select “Display store buffers”
  - Pick tile layer for a store to see which links are covered in 20 minute drive time.
  - For reference, turn on/off “Show all stores”

Display Store Buffers - MapViewer

- In Firefox, under Bookmarks, select “Display store buffers”
- Displays all the links that can reach Store 1
- Select different stores to view other buffers



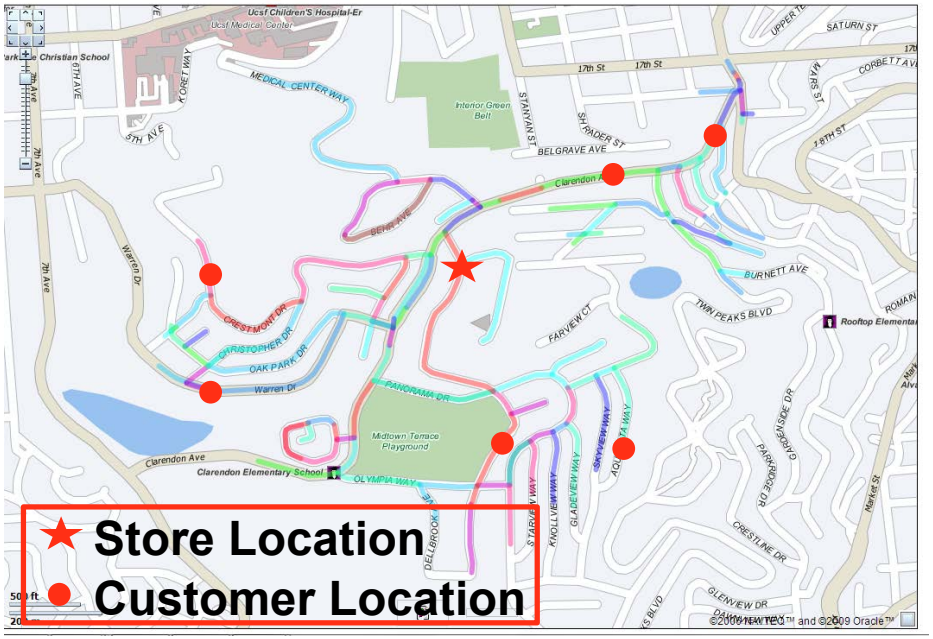
Find Each Customer's Closest Store



Large Scale Drive Time/Distance Analysis
Strategy

For millions of customers, find closest store within a specified drive time

- Same underlying data for geocoder and road network
- Customers geocode as link id and percentage (instead of longitude/latitude)
- 20 minute “reverse“ Network Buffer from each store generates all possible paths
- Each persisted path includes:
  - Covered link IDs, nodes ID, and associated costs
- Single database query to find closest store and drive time/distance for each customer (join on link_id)



 | |

Find Each Customer's Closest Store (and Cost)

- Same customer may fall on network buffer of multiple stores.
- Single SQL statement to generate closest store and cost of each customer.

Find Each Customer's Closest Store (and Cost)

- Connect to oracledemo user to run SQL.
- Results stored in results table.
  - cd WORKSHOP/SOURCE_CODE_EXAMPLE/ANALYSIS
  - oracledemo
  - SQL> @generate_final_results.sql
  - SQL> SELECT customer_id, store_id, cost_in_min
    FROM results
    WHERE rownum < 10;

Find a Customer That Can Reach Many Stores

- The following SQL will list all the stores customer 207 can reach in 20 minutes, ordered by cost.
  - SQL> @find_all_stores_for_one_cust.sql

C_CUSTOMER_ID	B_BUFFER_ID	COST
207	1	15.1884611
207	2	18.0122747
207	3	16.1757251
207	7	18.4145204
207	8	19.640776
207	9	19.9489666

- Only the closest store (store_id 1) will be in the results table

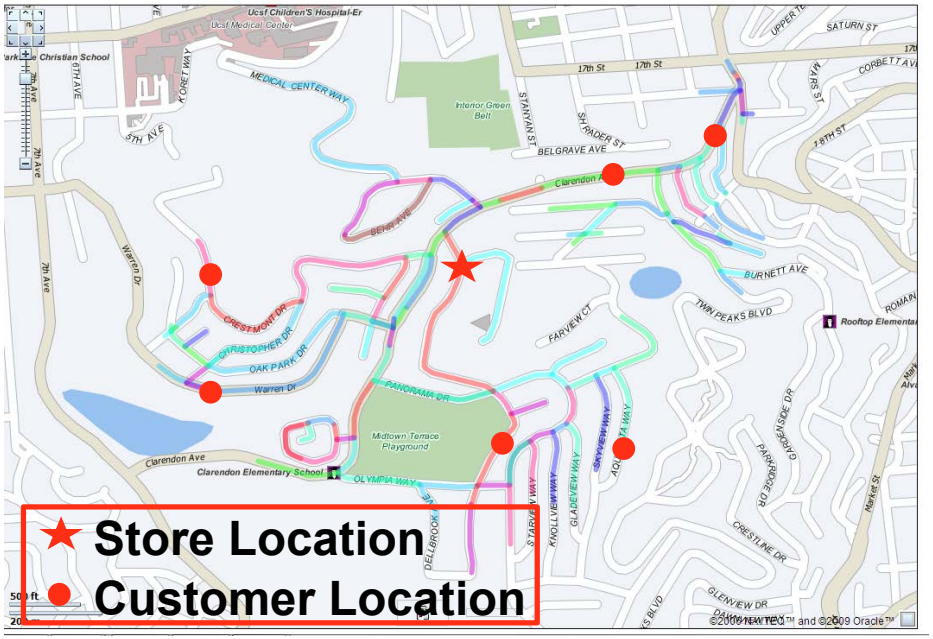
Conclusion

Large Scale Drive Time/Distance Analysis

Oracle Strategy

For millions of customers, find closest store within a specified drive time

- Same underlying data for geocoder and road network
- Customers geocode as link id and percentage (instead of longitude/latitude)
- 20 minute “reverse“ Network Buffer from each store generates all possible paths
- Each persisted path includes:
  - Covered link IDs, nodes ID, and associated costs
- Single database query to find closest store and drive time/distance for each customer (join on link_id)



Summary

- Drive time analysis can be more effective than as the crow flies
- No need to compute every customer / store drive time
- Think out of the box
  - Geocode customers to get link_id and percentage
  - Generate and persist network buffers for stores
  - Simple relational join to find closest customer to each store
- Easy and fast to perform additional analysis
  - Compute another set of store buffers (30 min drive time, or 20 miles away)
  - Run relational query (customers do not need to be re-geocoded).

D E M O N S T R A T I O N

NDM Tutorial – Demo

Source Code available here:

http://www.oracle.com/technetwork/indexes/samplecode/ndm-graph-1947612.html

Network Constraint Example

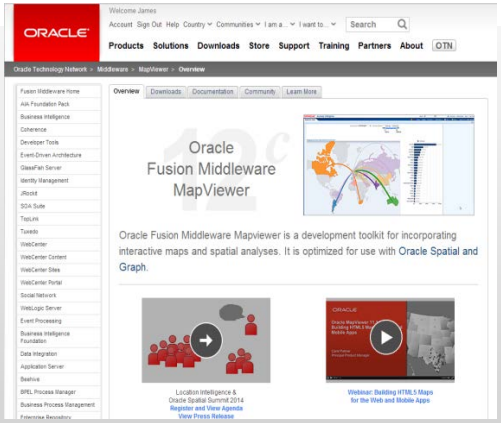
Customize Oracle's Algorithms

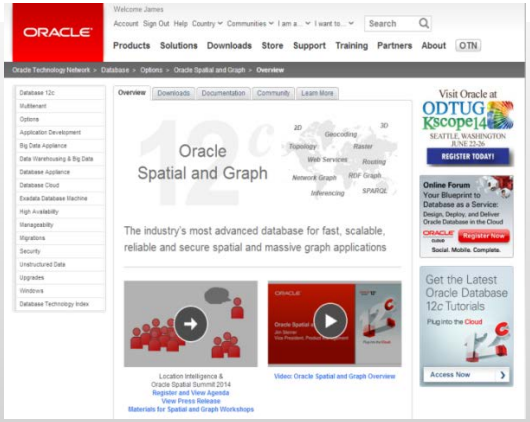

```
public class NoHighwayConstraint  implements LODNetworkConstraint
{
  public  NoHighwayConstraint(){}

 public boolean isSatisfied(LODAnalysisInfo info)
 {
   LogicalLink link = info.getNextLink();
   if (link==null || link.getLevel() == 1 )
     return true;
   else
     return false;
 }
```

Resources

Oracle Technology Network



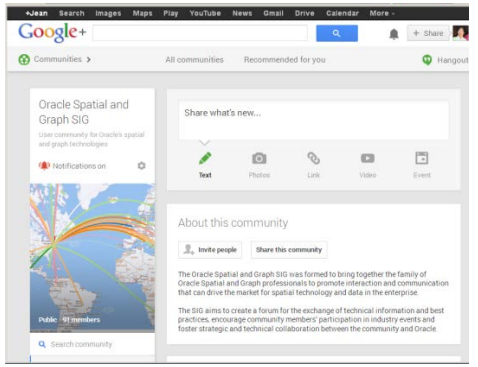


- www.oracle.com/technetwork/database/options/spatialandgraph
- www.oracle.com/technetwork/middleware/mapviewer
- https://blogs.oracle.com ➔ oraclespatial
  ➔ oracle_maps_blog

Oracle Spatial & Graph Special Interest Group

Connect and exchange knowledge with the community of Spatial & Graph users

- **Talk with the Board this week**
  - Wednesday lunch – SIG Board presentation (150AB)
  - Stop by the SIG User Group roundtable at Meet the Experts, 4:30pm Wednesday in150AB
  - Visit Oracle's exhibitor table at breaks & sign up



- **Join us**
  - Online communities: LinkedIn , Google+ , IOUG SIG (free membership)
  - Visit OTN Spatial Community page www.oracle.com/technetwork/database/options/spatialandgraph/community (or search online for **"Oracle Spatial and Graph Community"**)
  - Email oraclespatialsig@gmail.com

Spatial Certification & Partner Specialization

Get valuable credentials – differentiate your skills

- **Learn more at the Summit**
  - Wed, Track C 3:30 – Exam preparation session
  - Talk to us at Oracle's exhibitor table & "Meet the Experts" Certification table (Wed 4:30-5:00)
- **Take the next steps**
  - Schedule an exam, access topic lists / online training, learn about Partner Specialization requirements
    www.oracle.com/technetwork/database/options/spatialandgraph/learnmore/spatial-specialization-1835642.html
  - Online training materials for Certified Implementation Specialist exam
    https://competencycenter.oracle.com/opncc/full_glp.cc?group_id=22003

ORACLE

May 21, 2014
Walter E. Washington Convention Center
Washington, DC USA