



# Harnessing Location with Node.js and Oracle Database

**David Lapp**

Product Manager  
Oracle Spatial and Graph

**Siva Ravada**

Sr. Director of Development  
Oracle Spatial and Graph

Sept 16, 2019

## Safe Harbor

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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at <http://www.oracle.com/investor>. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

## Why This Matters

# Everything happens somewhere.

Enterprise and consumer apps are now expected to be “location-enabled”. With the maturity of geospatial features in common dev and data management platforms and libraries, it’s easy to harness location in your applications.

# Agenda

- Node.js and Oracle Spatial; concepts and drivers
- Demo 1
  - Build a REST API - Data fusion and proximity analysis
- Demo 2
  - Build a Web app – Location tracking and geofencing

# Node.js

- Open-source JavaScript run-time
- Designed for asynchronous, event driven applications
- Built to handle streaming content
- Easily create REST APIs
- Ecosystem of modules

# Oracle Database - Spatial and Graph

*Deployable Components*

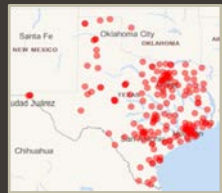
Mapping

Geocoding

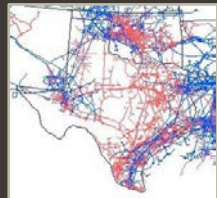
Routing

Web Services (OGC)

Studio



Points



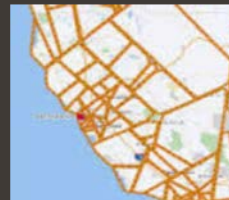
Lines



Polygons



Location Tracking  
(Geofencing)



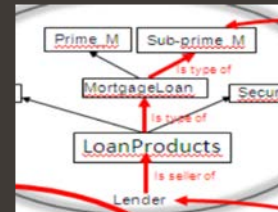
Networks



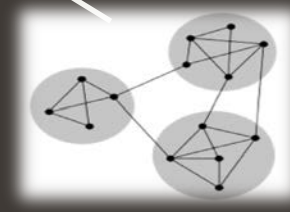
Raster



3D / LiDAR



RDF Graphs



Property Graphs



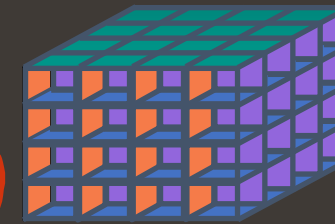
Topologies

# Oracle Spatial and Graph

## Native Geospatial Data Types



## Native Spatial Indexing



Fast Access to  
Spatial Data

ORACLE

## Spatial Analysis Through SQL

```
SELECT a.customer_name, a.phone_number
FROM policy_holders a
WHERE sdo_within_distance( a.geom, hurricane_path_geom,
                           'distance = 10 unit = mile') = 'TRUE';
```



# Oracle Spatial and Graph

## 20 Spatial Operators

- 20.1 SDO\_ANYINTERACT
- 20.2 SDO\_CONTAINS
- 20.3 SDO\_COVEREDBY
- 20.4 SDO\_COVERS
- 20.5 SDO\_EQUAL
- 20.6 SDO\_FILTER
- 20.7 SDO\_INSIDE
- 20.8 SDO\_JOIN
- 20.9 SDO\_NN
- 20.10 SDO\_NN\_DISTANCE
- 20.11 SDO\_ON
- 20.12 SDO\_OVERLAPBDYDISJOINT
- 20.13 SDO\_OVERLAPBDYINTERSECT
- 20.14 SDO\_OVERLAPS
- 20.15 SDO\_POINTINPOLYGON
- 20.16 SDO\_RELATE
- 20.17 SDO\_TOUCH
- 20.18 SDO\_WITHIN\_DISTANCE

## 21 Spatial Aggregate Functions

- 21.1 SDO\_AGGR\_CENTROID
- 21.2 SDO\_AGGR\_CONCAT\_LINES
- 21.3 SDO\_AGGR\_CONCAVEHULL
- 21.4 SDO\_AGGR\_CONVEXHULL
- 21.5 SDO\_AGGR\_LRS\_CONCAT
- 21.6 SDO\_AGGR\_MBR
- 21.7 SDO\_AGGR\_SET\_UNION
- 21.8 SDO\_AGGR\_UNION

## 26 SDO\_GEOM Package (Geometry)

- 26.1 SDO\_GEOM.RELATE
- 26.2 SDO\_GEOM.SDO\_ALPHA\_SHAPE
- 26.3 SDO\_GEOM.SDO\_ARC\_DENSIFY
- 26.4 SDO\_GEOM.SDO\_AREA
- 26.5 SDO\_GEOM.SDO\_BUFFER
- 26.6 SDO\_GEOM.SDO\_CENTROID
- 26.7 SDO\_GEOM.SDO\_CLOSEST\_POINTS
- 26.8 SDO\_GEOM.SDO\_CONCAVEHULL
- 26.9 SDO\_GEOM.SDO\_CONCAVEHULL\_BOUNDARY
- 26.10 SDO\_GEOM.SDO\_CONVEXHULL
- 26.11 SDO\_GEOM.SDO\_DIAMETER
- 26.12 SDO\_GEOM.SDO\_DIAMETER\_LINE
- 26.13 SDO\_GEOM.SDO\_DIFFERENCE
- 26.14 SDO\_GEOM.SDO\_DISTANCE
- 26.15 SDO\_GEOM.SDO\_ENDPOINTS
- 26.16 SDO\_GEOM.SDO\_LENGTH
- 26.17 SDO\_GEOM.SDO\_MAX\_MBR\_ORDINATE
- 26.18 SDO\_GEOM.SDO\_MAXDISTANCE
- 26.19 SDO\_GEOM.SDO\_MAXDISTANCE\_LINE
- 26.20 SDO\_GEOM.SDO\_MBC
- 26.21 SDO\_GEOM.SDO\_MBC\_CENTER
- 26.22 SDO\_GEOM.SDO\_MBC\_RADIUS
- 26.23 SDO\_GEOM.SDO\_MBR
- 26.24 SDO\_GEOM.SDO\_MIN\_MBR\_ORDINATE
- 26.25 SDO\_GEOM.SDO\_POINTONSURFACE
- 26.26 SDO\_GEOM.SDO\_SELF\_UNION
- 26.27 SDO\_GEOM.SDO\_TRIANGULATE
- 26.28 SDO\_GEOM.SDO\_UNION
- 26.29 SDO\_GEOM.SDO\_VOLUME

## 30 SDO\_PC\_PKG Package (Point Clouds)

- 30.1 SDO\_PC\_PKG.CLIP\_PC
- 30.2 SDO\_PC\_PKG.CLIP\_PC\_FLAT
- 30.3 SDO\_PC\_PKG.CREATE\_CONTOUR\_GEOMETRIES
- 30.4 SDO\_PC\_PKG.CREATE\_PC
- 30.5 SDO\_PC\_PKG.DROP\_DEPENDENCIES
- 30.6 SDO\_PC\_PKG.GET\_PT\_IDS
- 30.7 SDO\_PC\_PKG.HAS\_PYRAMID
- 30.8 SDO\_PC\_PKG.INTERSECT\_PC
- 30.9 SDO\_PC\_PKG.PC2DEM
- 30.10 SDO\_PC\_PKG.PRESERVES\_LEVEL1
- 30.11 SDO\_PC\_PKG.SDO\_PC\_NN
- 30.12 SDO\_PC\_PKG.SDO\_PC\_NN\_FOR\_EACH

## 31 SDO\_SAM Package (Spatial Analysis and Mining)

- 31.1 SDO\_SAM.AGGREGATES\_FOR\_GEOMETRY
- 31.2 SDO\_SAM.AGGREGATES\_FOR\_LAYER
- 31.3 SDO\_SAM.BIN\_GEOMETRY
- 31.4 SDO\_SAM.BIN\_LAYER
- 31.5 SDO\_SAM.COLOCATED\_REFERENCE\_FEATURES
- 31.6 SDO\_SAM.SIMPLIFY\_GEOMETRY
- 31.7 SDO\_SAM.SIMPLIFY\_LAYER
- 31.8 SDO\_SAM.SPATIAL\_CLUSTERS
- 31.9 SDO\_SAM.TILED\_AGGREGATES
- 31.10 SDO\_SAM.TILED\_BINS

## 27 SDO\_LRS Package (Linear Referencing System)

- 27.1 SDO\_LRS.CLIP\_GEOM\_SEGMENT
- 27.2 SDO\_LRS.CONCATENATE\_GEOM\_SEGMENTS
- 27.3 SDO\_LRS.CONNECTED\_GEOM\_SEGMENTS
- 27.4 SDO\_LRS.CONVERT\_TO\_LRS\_DIM\_ARRAY
- 27.5 SDO\_LRS.CONVERT\_TO\_LRS\_GEOM
- 27.6 SDO\_LRS.CONVERT\_TO\_LRS\_LAYER
- 27.7 SDO\_LRS.CONVERT\_TO\_STD\_DIM\_ARRAY
- 27.8 SDO\_LRS.CONVERT\_TO\_STD\_GEOM
- 27.9 SDO\_LRS.CONVERT\_TO\_STD\_LAYER
- 27.10 SDO\_LRS.DEFINE\_GEOM\_SEGMENT
- 27.11 SDO\_LRS.DYNAMIC\_SEGMENT
- 27.12 SDO\_LRS.FIND\_LRS\_DIM\_POS
- 27.13 SDO\_LRS.FIND\_MEASURE
- 27.14 SDO\_LRS.FIND\_OFFSET
- 27.15 SDO\_LRS.GEOM\_SEGMENT\_END\_MEASURE
- 27.16 SDO\_LRS.GEOM\_SEGMENT\_END\_PT
- 27.17 SDO\_LRS.GEOM\_SEGMENT\_LENGTH
- 27.18 SDO\_LRS.GEOM\_SEGMENT\_START\_MEASURE
- 27.19 SDO\_LRS.GEOM\_SEGMENT\_START\_PT
- 27.20 SDO\_LRS.GET\_MEASURE
- 27.21 SDO\_LRS.GET\_NEXT\_SHAPE\_PT
- 27.22 SDO\_LRS.GET\_NEXT\_SHAPE\_PT\_MEASURE
- 27.23 SDO\_LRS.GET\_PREV\_SHAPE\_PT
- 27.24 SDO\_LRS.GET\_PREV\_SHAPE\_PT\_MEASURE

- 100's of spatial operators and functions
- From basic to advanced
- From general purpose to specialized



# Spatial data type

```
SQL> desc countries
```

```
Name          Null? Type
```

```
-----
```

```
ID            NUMBER
```

```
ISO_A3        VARCHAR2(3)
```

```
NAME          VARCHAR2(26)
```

```
GEOMETRY      MDSYS.SDO_GEOMETRY
```

```
SQL>
```

```
SQL> SELECT geometry
```

```
2 FROM countries
```

```
3* WHERE name='Aruba';
```

```
GEOMETRY
```

```
-----
```

```
SDO_GEOMETRY(2003, 8307, NULL,  
SDO_ELEM_INFO_ARRAY(1, 1003, 1),  
SDO_ORDINATE_ARRAY(-69.8760919, 12.42720123, -  
69.879425, 12.45340118, -69.9150301,  
12.49686106, -69.9238926, 12.51903025, -  
69.935649, 12.5316393, -69.9961879,  
12.57737295, ...
```

# Spatial query

```
SQL> SELECT a.name  
2 FROM populated_places a, countries b  
3 WHERE sdo_inside(a.geometry, b.geometry) = 'TRUE'  
4* and b.name='Belize';
```

NAME

-----

El Cayo  
Punta  
Gorda  
Belmopan  
Orange  
...

# Spatial query

```
SQL> SELECT name
      2 FROM countries
      3 WHERE sdo_contains(
      4         geometry,
      5         SDO_GEOMETRY(2001,8307,
      6             SDO_POINT_TYPE(-99.3, 23.1, NULL), NULL, NULL))
      7         = 'TRUE';
```

NAME

-----  
Mexico

# Spatial cast to/from GeoJSON

```
SQL> SELECT sdo_util.to_geojson(geometry)
2 FROM countries
3* WHERE name='Aruba';
```

```
SDO_UTIL.TO_GEOJSON(GEOMETRY)
```

```
-----
{ "type": "Polygon", "coordinates": [ [ [-69.8760918785688,
12.4272012328116], [-69.8794250088263, 12.4534011841892],
[-69.9150300699863, 12.4968610642473], [-69.923892578319,
12.5190302533902], [-69.9356489664463, 12.5316393031227],
[-69.9961879071357, 12.5773729453627], [-70.006368164159,
12.5853827920903], [-70.0480452075418,
12.6319949343567],... ] ] }
```

# Spatial cast to/from GeoJSON

```
SQL> SELECT name
      2 FROM countries
      3 WHERE sdo_contains(
      4         geometry,
      4         sdo_util.from_geojson(
      5             '{"type": "Point", "coordinates": [-99.3, 23.1] }'))
      6         = 'TRUE';
```

NAME

-----

Mexico

# Supply fire perimeter as GeoJSON→ find nearby sites

```
SQL> SELECT facility_name,  
2         round(SDO_NN_DISTANCE(1),1) distance,  
3         sdo_util.to_geojson(geometry) as geometry_gj  
4 FROM   demo_data.epa_tri_facilities  
5 WHERE  sdo_nn(geometry,sdo_util.from_geojson(  
6         '{"type": "Polygon", "coordinates":[[[-110.497, 37.658],[-110.493, 37.658],'|'|  
7         '[-110.493, 37.660],[-110.497, 37.660],[-110.497, 37.658]]]}''),  
8         'sdo_num_res=10 distance=500 unit=KM', 1) ='TRUE'  
9 ORDER BY distance;
```

| FACILITY_NAME                 | DISTANCE | GEOMETRY_GJ              |
|-------------------------------|----------|--------------------------|
| SALT RIVER GENERATING STATION | 115.2    | { "type": "Point", ... } |
| LISBON VALLEY MINING CO       | 125.1    | { "type": "Point", ... } |
| PACIFICORP HUNTER             | 174.5    | { "type": "Point... }    |
| US GYPSUM CO                  | 183.7.   | { "type": "Point",... }  |
| ...                           |          |                          |



# node-oracledb

- Node.js add-on for robust access to Oracle db
- Open source, maintained by Oracle
- Oracle Node.js Developer Center

<https://www.oracle.com/database/technologies/appdev/nodejs.html>

- Doc, examples

<https://oracle.github.io/node-oracledb/>

- Source

<https://github.com/oracle/node-oracledb>

- Async/Await, Promises, Callbacks and Streams
- SQL and PL/SQL execution
- Oracle Database High Availability features
- Oracle Net features including encryption
- Array Fetches and Bulk loading features
- Oracle Named Types and Collection support
- Large Objects: CLOBs and BLOBs as Streams or Strings and Buffers
- Oracle Database 12c JSON datatype
- Simple Oracle Document Access (SODA)
- Continuous Query Notification (CQN)
- Advanced Queuing (AQ)
- REF CURSORS and Implicit Results
- Data binding using JavaScript objects or arrays
- Inbuilt Connection Pool with Queuing, Aliasing, Tagging, Draining, Heterogeneous and Homogeneous connections, Proxy connections and Liveness checking
- Database Resident Connection Pooling (DRCP)
- Privileged connections
- External authentication
- Password changing
- Statement Saching and Client Result Caching
- End-to-end tracing, mid-tier authentication, and auditing

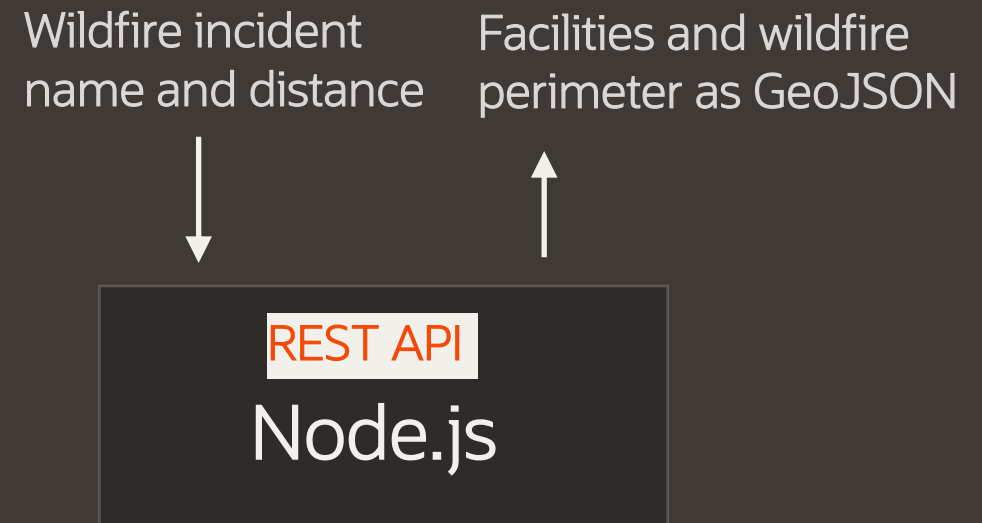
# Why Node.js with Spatial?

- ✓ Fusion of internal/external location-based data/services
- ✓ Subscribe to location-based events and streams
- ✓ JavaScript and GeoJSON pervasive in web mapping apps
- ✓ Robust Node.js module ecosystem
  - node-oracledb driver maintained by Oracle
  - Express for REST API
  - Socket.IO for event handling
- ✓ Cloud-ready
  - Node.js on Oracle Container Engine for Kubernetes or OCI Compute
  - Spatial on ExaCS, ExaCC, ADW, ATP, DBCS High/Extreme Performance

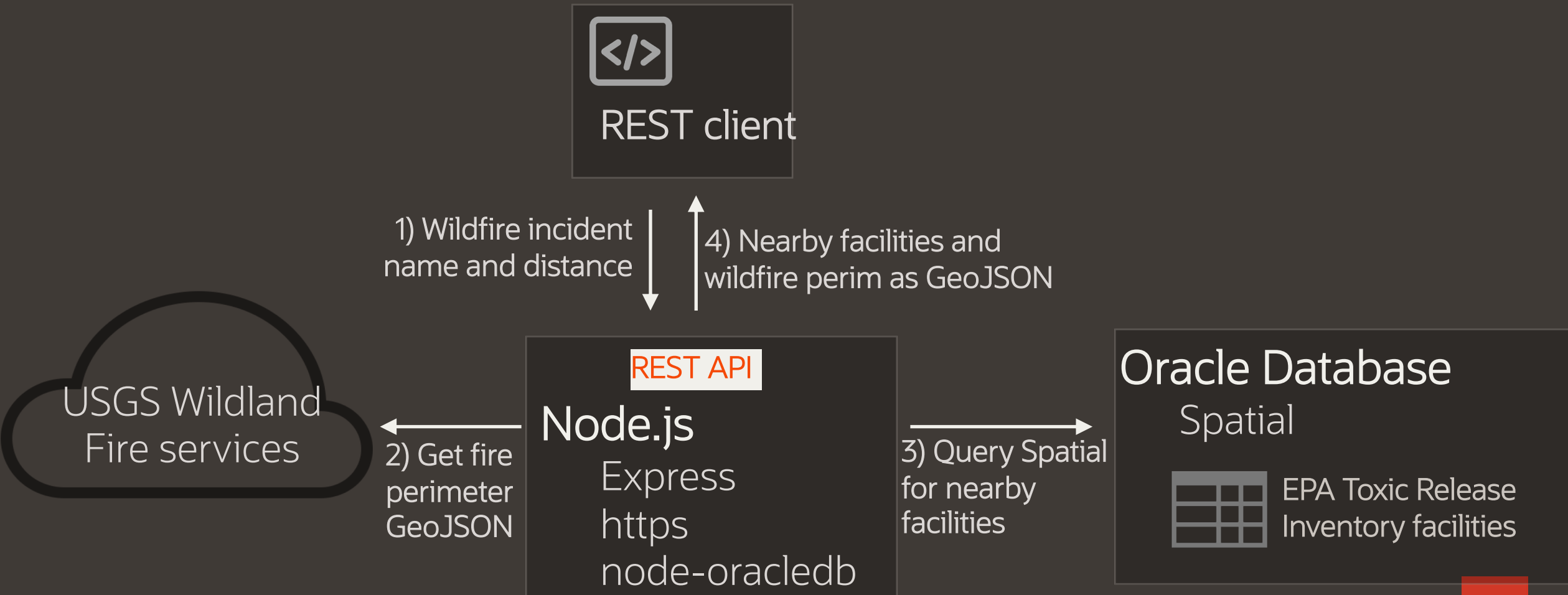
# Demo 1:

## REST API - Data fusion and proximity analysis

- Wildfire info from external geospatial web service
- Facility info and spatial analysis from Oracle Spatial



# Demo 1: REST API - Data fusion and proximity analysis



# Demo 1

# Demo 1: Code walkthrough

*//adapted from <https://github.com/oracle/node-oracledb/tree/master/examples>*

*... require modules express, https, oracledb ...  
... proxy and oracledb settings ...*

```
const app = express();
```

```
const server = app.listen(3000, () => {  
  var host = server.address().address  
  var port = server.address().port  
  console.log('app listening at http://%s:%s', host, port)  
})
```

```
app.get('/incidentname/:incidentname/distance/:distance', (req, res) => {  
  getFirePerim(req, res)  
})
```

## Demo 1: Code walkthrough cont.

```
const options = ... USGS service host, port, method...
const path = ... USGS service fixed path elements...

const getFirePerim = (req, res) => {
  console.log('getting fire perimeter from USGS service...')
  options.path = path + '%27' + req.params.incidentname + '%27'
  https.request(options, (res2) => {
    let body = ''
    res2.on('data', (chunk) => {
      body += chunk
      console.log(' get chunk...')
    })
    res2.on('end', () => {
      if (JSON.parse(body).features.length > 0) {
        const gj = JSON.stringify(JSON.parse(body).features[0].geometry)
        runQuery(gj, req, res)
      } else { res.send('cannot get fire perimeter') }
    })
  }).end()
}
```

## Demo 1: Code walkthrough cont.

```
async function runQuery(gj, req, res){
  const gjArray = []
  try {
    // Dedicated connection for demo. For a production app use a connection pool.
    const connection = await oracledb.getConnection(dbConfig)
    const nnParams = 'sdo_num_res=100 distance=' + req.params.distance + ' unit=KM';
    const result = await connection.execute(
      `select facility_name, industry_sector, round(SDO_NN_DISTANCE(1),1) distance,
        sdo_util.to_geojson(geometry) as geometry
        from epa_tri_facilities
        where sdo_nn(geometry,sdo_util.from_geojson(:1),
          '${nnParams} unit=KM', 1) = 'TRUE'
        order by distance`, { 1: { val: gj, type: oracledb.CLOB } })

    result.rows.forEach(function (item) {
      const tmp = { type: 'Feature' }
      tmp.properties = { facility_name: item.FACILITY_NAME, distance: item.DISTANCE }
      tmp.geometry = JSON.parse(item.GEOMETRY)
      gjArray.push(tmp)
    })
  }
```



# Demo 1: Code walkthrough cont.

```
const perim = { type: 'Feature' }
perim.properties = { incidentname: req.params.incidentname }
perim.geometry = JSON.parse(gj)
gjArray.push(perim)

const geojson = { type: 'FeatureCollection', features: gjArray }
res.setHeader('Content-Type', 'application/json')
res.send(
  JSON.stringify(geojson))
console.log('... done')
```

# Demo 2:

## Web app – Location tracking and geofencing

Context...

- Geofences are predefined regions that represent areas of interest, such as the regions surrounding secure facilities
- Location tracking detects objects entering/exiting a geofence and sends messages
- Oracle Spatial Location Tracking Server
  - In-database feature of Oracle Spatial and Graph
  - Uses Oracle Database Advanced Queuing (AQ) for event messages
  - Designed for large volume of complex geofences and many moving objects
  - For simplicity, in this demo we track only 1 object

# Demo 2:

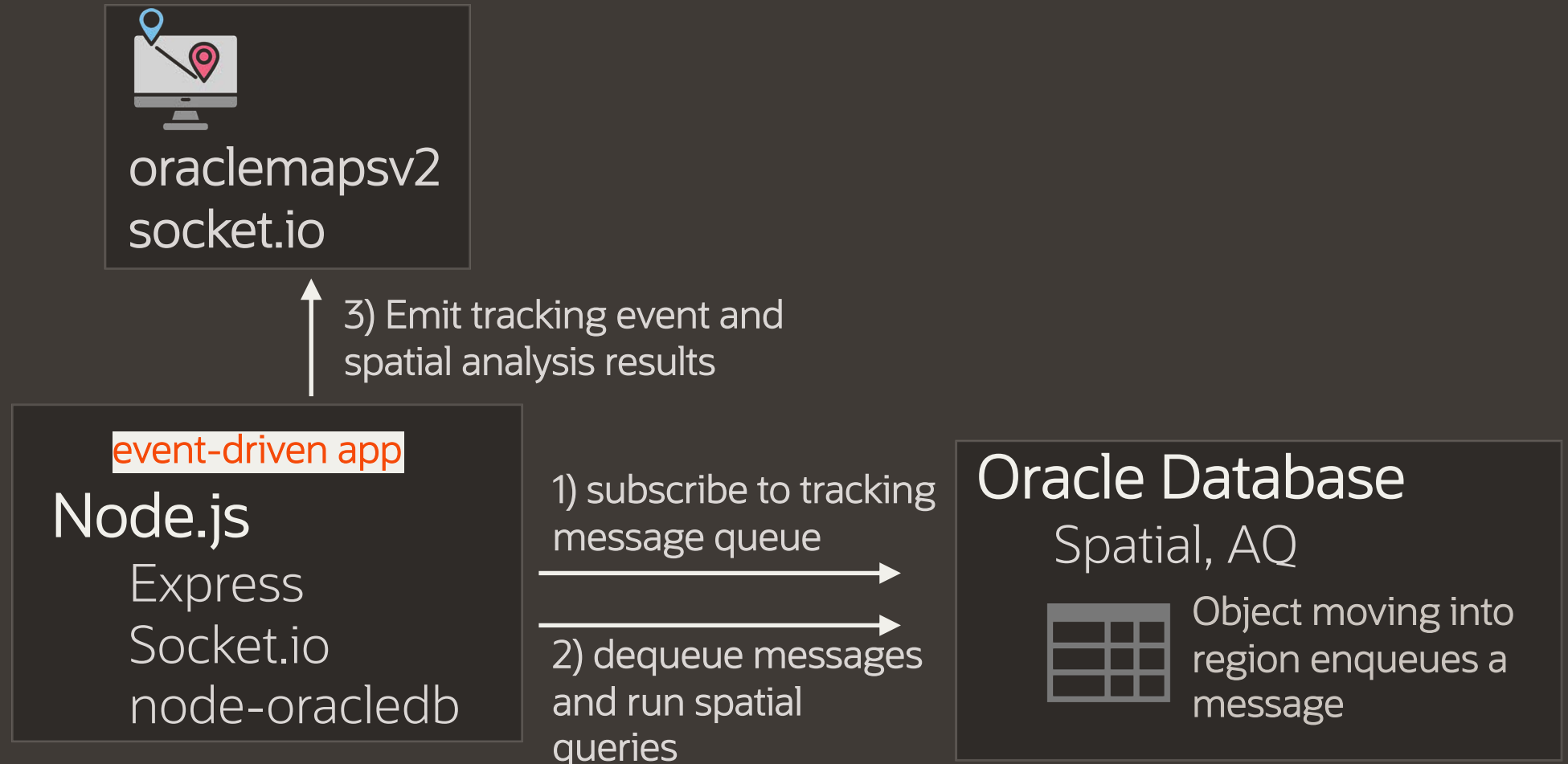
## Web app – Location tracking and geofencing

- An object continually moves to random locations
- Location and geofencing messages from Oracle Spatial Location Tracking Server
- Spatial analysis results from Oracle Spatial queries



# Demo 2:

## Web app – Location tracking and geofencing



# Demo 2

## Demo 2: Server code walkthrough

```
// adapted from examples at
// https://github.com/oracle/node-oracledb/tree/master/examples

... require modules express, http, socket.io, oracledb ...
... proxy and oracledb settings ...

async function startExpress () {
  app.get('/', function (req, res) {
    res.sendFile(__dirname + '/socket-trkr-map.html')
  })
  app.post('/countries.json', function (req, res) {
    res.sendFile(__dirname + '/countries.json')
  })
  http.listen(port, function () {
    console.log('Listening on http://localhost:' + port)
  })
}
```

## Demo 2: Server code walkthrough cont.

```
const query =
`WITH x as (
  SELECT sdo_geometry(2001,8307, sdo_point_type(:1, :2, null),null,null) as g
  FROM dual )
  SELECT a.name, a.iso_a3 as iso_code,
         round(sdo_geom.sdo_distance(
             x.g,
             sdo_util.polygontoline(a.geometry),
             0.05, 'unit=KM'),0) as dist_from_border_km,
         b.nameasci||', '||b.adm0name as nearest_populated_place,
         round(sdo_nn_distance(1)/1000,2) as distance_from_nearest_place
  FROM countries_oceans a, ne_10m_populated_places_simple b , x
  WHERE id=:3
  AND sdo_nn(b.geometry, x.g, 'sdo_num_res=1',1)='TRUE'`;
```

## Demo 2: Server code walkthrough cont.

```
async function startOracle() {
  let connection;
  try {
    let result;
    // Dedicated connection for demo. For a production app use a connection pool.
    connection = await oracledb.getConnection(dbConfig);
    const queue = await connection.getQueue(queueName,
                                             {payloadType: "MDSYS.NOTIFICATION_MSG"});
    const msg = await queue.deqOne(); // wait for a message
    await connection.commit();
    if (msg) {
      result = await connection.execute(
        query, [msg.payload.X, msg.payload.Y, msg.payload.REGION_ID] );
      io.emit('message', [msg.payload, result.rows]);
    }
    setTimeout(function(){startOracle();}, 3000);
  }
}
```



## Demo 2: Client code walkthrough

```
... include jquery, oraclemapsv2, socket.io js...  
... create map and message divs...
```

```
$(function () {  
    var map = new OM.Map(document.getElementById('map'));  
    var markerLayer = new OM.layer.MarkerLayer("markerlayer1");  
    var insertMapMarker = function(id, cx, cy, srid, text) {  
        var myobj = {id: id, markerText: text,  
                    position: {'x': cx, 'y': cy, 'srid': srid}};  
        var mm = new OM.MapMarker(myobj);  
        markerLayer.addMapMarker(mm);  
    };  
    var countriesLayer = new OM.layer.VectorLayer('countries',  
        {def:{type:OM.layer.VectorLayer.TYPE_DATAPACK,  
            url:"http://localhost:3000/countries.json"}} );  
    countriesLayer.setRenderingStyle(new OM.style.Color(  
        {strokeThickness:1,stroke:"#f7fcb9",fill:"#b2bec6",fillOpacity:0.25}));  
});
```

## Demo 2: Client code walkthrough cont.

```
map.addLayer(countriesLayer);
map.addLayer(markerLayer);
map.addLayer(new OM.layer.ElocationTileLayer("tilelayer"));
map.setMapCenter( new OM.geometry.Point(-30, 40, 4326) );
map.setMapZoomLevel(1) ;
map.init() ;

var socket = io();
socket.on('message', function(msg){
    $('#messages').empty();
    $('#messages').append($('- ').text('Object ' + msg[0].OBJECT_ID));
    $('#messages').append($('- ').text(msg[0].TIME));
    $('#messages').append($('- ').text(msg[0].X+'°', ' '+msg[0].Y+'°'));
    $('#messages').append($('- ').text('-----'));
    $('#messages').append($('- ').text(msg[1][0].NAME));
    $('#messages').append($('- ').text(msg[1][0].DIST_FROM_BORDER_KM...));
    $('#messages').append($('- ').text(msg[1][0].DISTANCE_FROM_NEAREST_PLACE...));
    markerLayer.removeAllFeatures();
    insertMapMarker(0, msg[0].X, msg[0].Y, 4326, msg[0].OBJECT_ID);
});
});

```

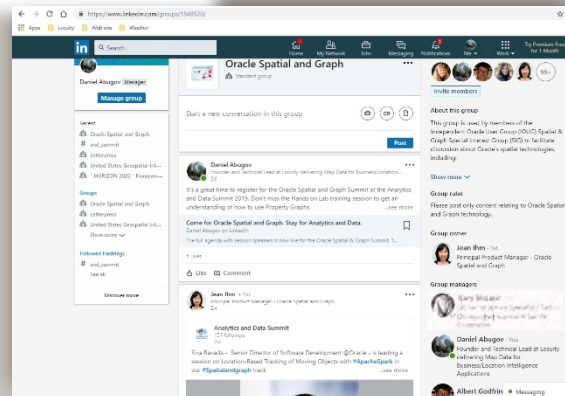
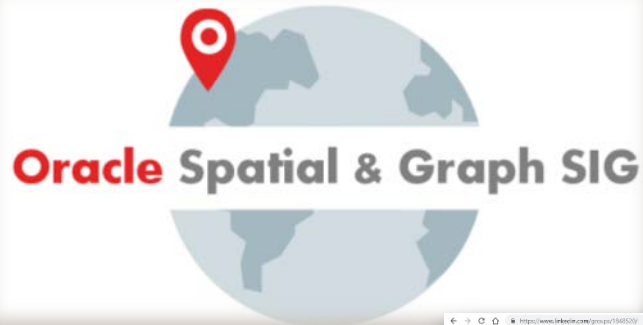
# Takeways

- It's easy to use the Spatial features of Oracle Database with Node.js apps
- Data fusion and REST API development are low hanging fruit
- node-oracledb driver greatly simplifies the process

# The Spatial & Graph SIG User Community

## *Now part of BIWA User Group*

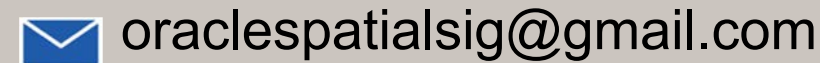
We are a vibrant community of customers and partners that connects and exchanges knowledge online, and at conferences and events.



Meet us at OpenWorld! Monday-Wednesday  
**Moscone West, Level 3, User Group area**  
at the *BIWA/Analytics Community* table

Join us online

[tinyurl.com/oraclespatialcommunity](https://tinyurl.com/oraclespatialcommunity)



[oraclespatialsig@gmail.com](mailto:oraclespatialsig@gmail.com)



The banner features a large photograph of a white building with a red-tiled roof and a central clock tower, set against a clear blue sky. In the foreground, there is a fountain with water spraying upwards. The text is overlaid on the right side of the image.

 **Analytics and Data Summit**

★ TechCasts   Submit Your Abstract   Become a Sponsor   News   Past Summits ▾   🔍

# SAVE THE DATE

# ANALYTICS AND DATA SUMMIT 2020

All Analytics. All Data.  
No Nonsense.

February 25-27, 2020

Call for Speakers Now Open!

[SIGN UP FOR OUR NEWSLETTER](#)

Formerly the BIWA Summit with the Spatial and Graph Summit.

[@AnalyticAndData](#) 

**analyticsanddatasummit.org**

**Seeking customer use cases and technology sessions**

**Dedicated Spatial & Graph track with 20+ sessions**

# Spatial at OOW and Code One 2019



Sessions, workshops, demos...  
**[bit.ly/SpatialGraphOOW19](https://bit.ly/SpatialGraphOOW19)**

ORACLE  
OPEN  
WORLD

September 16-19, 2019  
MOSCONE CENTER | SAN FRANCISCO

Content Highlights Sponsor/Exhibit Attend Register →

## PROGRAM GUIDE

### Oracle Spatial and Graph

#### Sessions

Customer Case Study Sessions

**All Analytics, All Data: No Nonsense** ☆

[Shyam Varan Nath](#), Director IoT & Cloud, BIWA User Group  
[Dan Vlamis](#), CEO - President, Vlamis Software Solutions, Inc.  
[Charlie Berger](#), Sr. Director Product Management, Machine Learning, AI and Cognitive Analytics, Oracle

**SCHEDULE** Monday, Sep 16, 9:00 a.m. - 9:45 a.m. | Moscone West - Room 3016

**Using Graph Analysis and Fraud Detection in the Fintech Industry** ☆

[Yavor Ivanov](#), DBA manager, Paysafe  
[Stanka Dalekova](#), Senior Software Engineer, Paysafe  
[Dobroslav Hristov](#), Senior Software Engineer, Paysafe

**SCHEDULE** Tuesday, Sep 17, 12:30 p.m. - 1:15 p.m. | Moscone South - Room 152C

# Resources - Get Started



Oracle Spatial and Graph product pages

[oracle.com/technetwork/database/options/spatialandgraph](https://oracle.com/technetwork/database/options/spatialandgraph)



YouTube channel [youtube.com/c/OracleSpatialandGraph](https://youtube.com/c/OracleSpatialandGraph)



Blog – examples, tips & tricks

[blogs.oracle.com/oraclespatial](https://blogs.oracle.com/oraclespatial)

| [blogs.oracle.com/bigdataspatialgraph](https://blogs.oracle.com/bigdataspatialgraph)

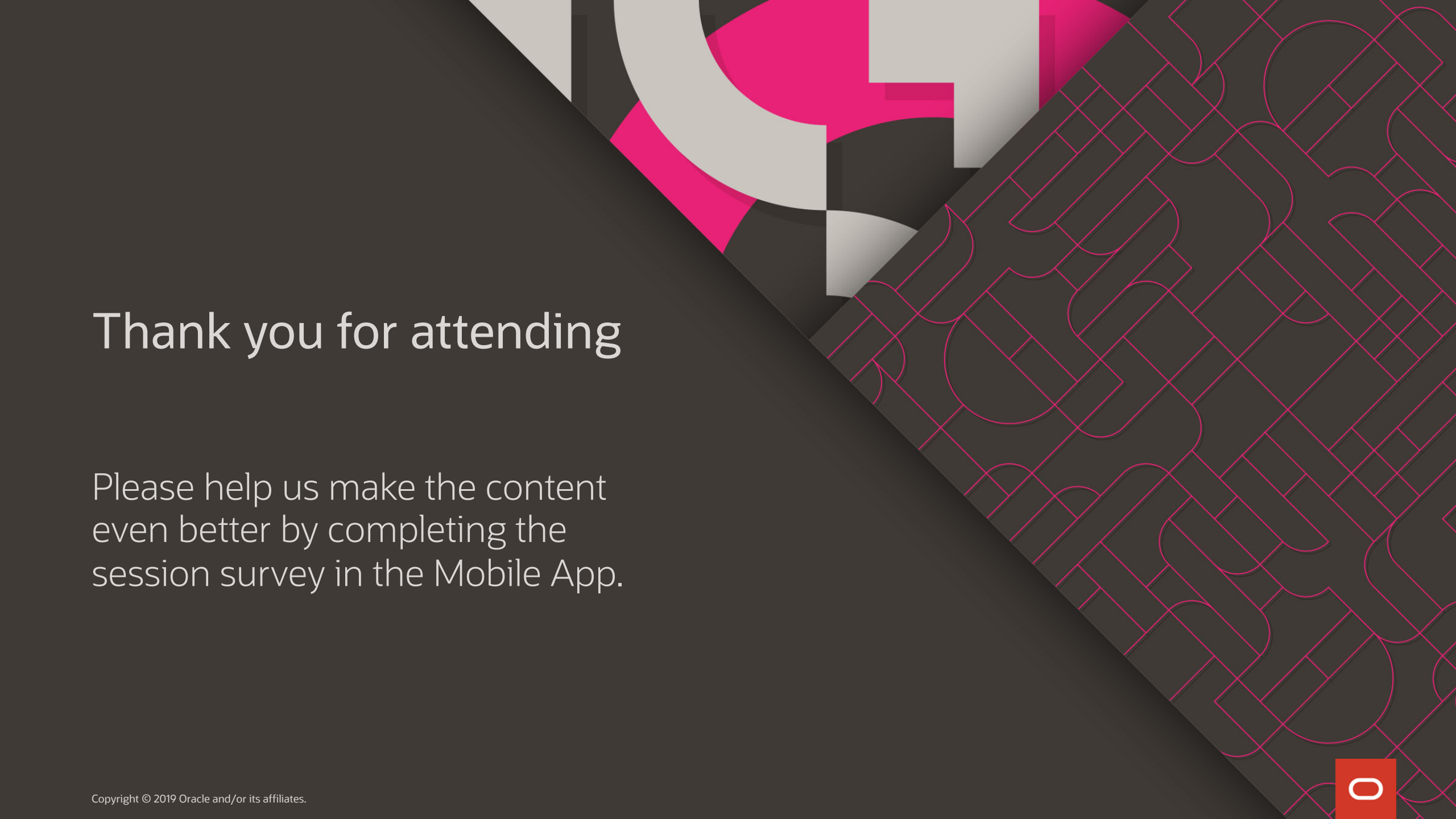


[@SpatialHannes](https://twitter.com/SpatialHannes)



[Oracle Spatial and Graph Group](https://www.linkedin.com/groups/Oracle-Spatial-and-Graph-Group-11071111)





# Thank you for attending

Please help us make the content even better by completing the session survey in the Mobile App.