ORACLE

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

How to Build a Better GIS Application

Siva Ravada
Senior Director of Development
Spatial and Graph & MapViewer
Oracle

Program Agenda

- A Land Management Application
- Building New Geometry Functions
- Automated Map Simplification

Important Topics covered

- Data Model based database design
- Trigger based spatial constraints
- Polygon snapping techniques
- Topologically consistent Map Generalization
- Data publishing with MapViewer

System Architecture

Data input in file format (from local government agencies)

GDAL

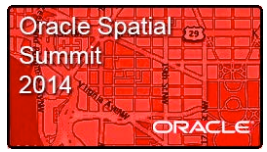
PL/SQL Procedures

Java Stored Procedures

Web Services

National Maps at different scales

System Input



- The data for the system is supplied in large files by various local government institutions
- Local Government institutions maintain their own data
  - This will result in small differences due to different standards for data collection
  - These differences cause many small overlaps and gaps in the system
- One of the main tasks of the system is to collect the data from various government institutions into a seamless map
  - In this process geometries are checked for overlaps and gaps and are corrected with an automated process

Data Publishing System

- Input HTTP request
  - Region of interest
  - Map Scale
- Output
  - XML response file with the map data
  - GeoJSON response file with map data
  - WFS
  - WMS

Define a Data Model

- A spatial database should be designed just like any other database with a fully specified model
  - A fully specified model that is application independent should control spatial data storage
  - A good data model supports and enhances data access without compromising quality
- Database features should be used to support limitations of specific applications rather than limiting the data model
  - For example, some applications mandate a single spatial column per table or only a single homogeneous geometry type per spatial column
    - These limitations can be accommodated using database features like views and triggers

Use database check constraints

- Use LAYER_GTYPE to constrain the feature types stored in a table
  - Very important to do this to avoid problems in the applications
- Spatial constraints can be created using PL/SQL code
  - A spatial constraint is a data consistency check that makes sure the data stored in the spatial layers follows certain spatial rules
  - For example, it may be that a building footprint should always be contained within a land parcel or a road should never cross a land parcel
  - The most common way to enforce this is to define triggers on the tables and check for data consistency as soon as a new row is inserted
- If the constraints are implemented at the DB level, all applications accessing the data will share the same data quality checks

Examples of Spatial constraints

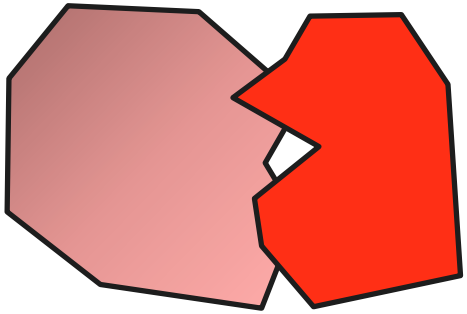
- CONTAINS: A land parcel may contain a building footprint
- COVERS: A planning neighbourhood covers a land parcel
- EQUAL: A planning neighbourhood can be equal to a land parcel
- TOUCH: A road segment can only touch another road segment
- CONTAINS+COVERS: A land parcel may contain and cover a building footprint
- Complex rules: Two land parcels may not overlap and there should be no gaps between adjacent land parcels
- All of these can be implemented in the DB using PL/SQL code and existing spatial functions

Implementing New Functions

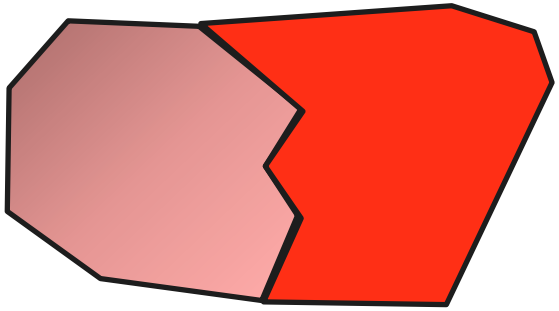
- SDO_GEOMETRY is an open type and easy to understand
- Existing functionality in the DB can be extended using PL/SQL or Java stored procedures
- Example: Swapping ordinates in the geometry
  - Data comes in as lat/long, but SDO_GEOMETRY expects it as long/lat
  - Does the client need to do the conversion ?
  - Can be easily done in PL/SQL on insert into the table

Detecting overlaps between polygons

- SDO_GEOM.RELATE function used to detect if there is an overlap between two polygons
  - If the relationship between them is TOUCH there is no overlap
    - There may be gaps between them

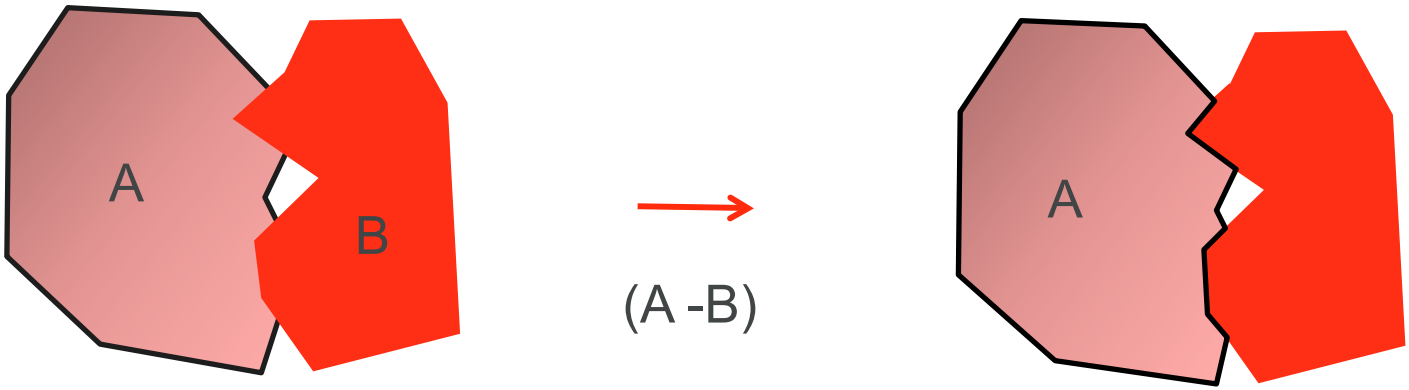


Overlap



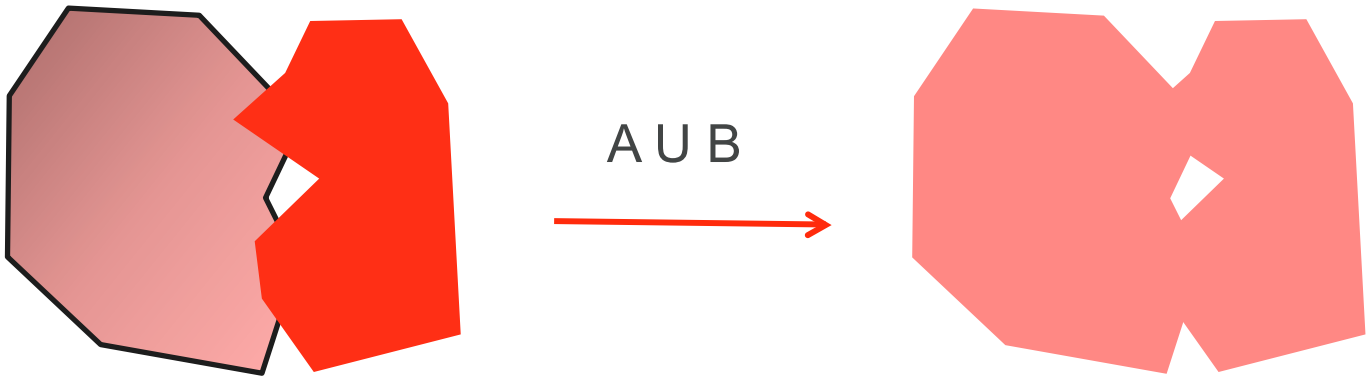
No Overlap

Overlap Correction



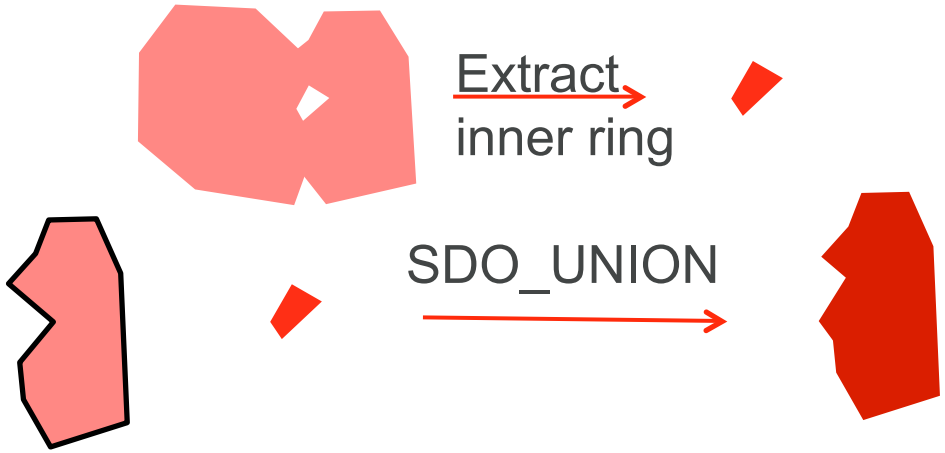
Detecting gaps between polygons

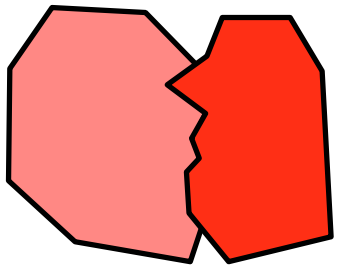
- Take the outer rings of polygons A and B
- SDO_UNION both the outer rings
  - If the resulting polygon has holes (interior rings) then there are gaps between the two input polygons



Fixing gaps between polygons

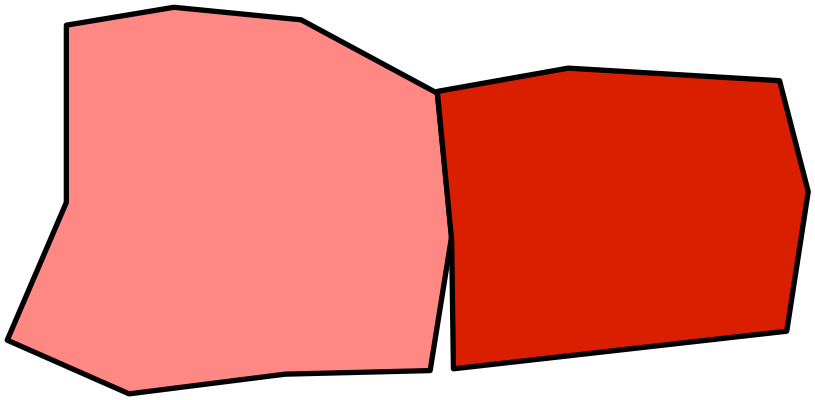
- Extract the inner rings and add them to Polygon B
  - SDO_UTIL.EXTRACT function can be used in the DB
  - SDO_UNION of inner ring with Polygon B





Final Result

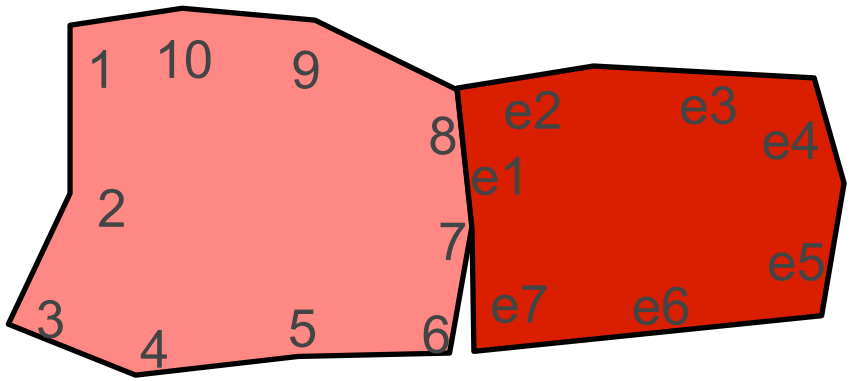
Detecting gaps between polygons



- In this situation the SDO_UNION method with rings does not work
- SDO_GEOM.RELATE will say TOUCH
  - But there are gaps that need to be fixed

Detecting boundary gaps

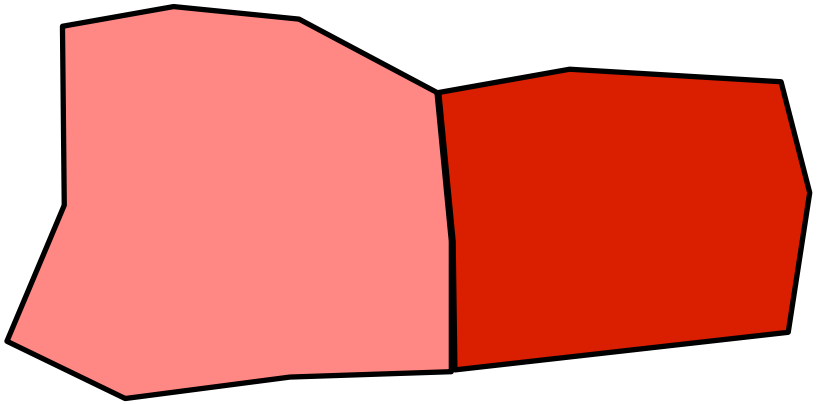
- Explode geometry B into line segments
- If vertices from geometry A are within 1 meter from edges of geometry B (but not on the edge) then there are gaps between the polygons



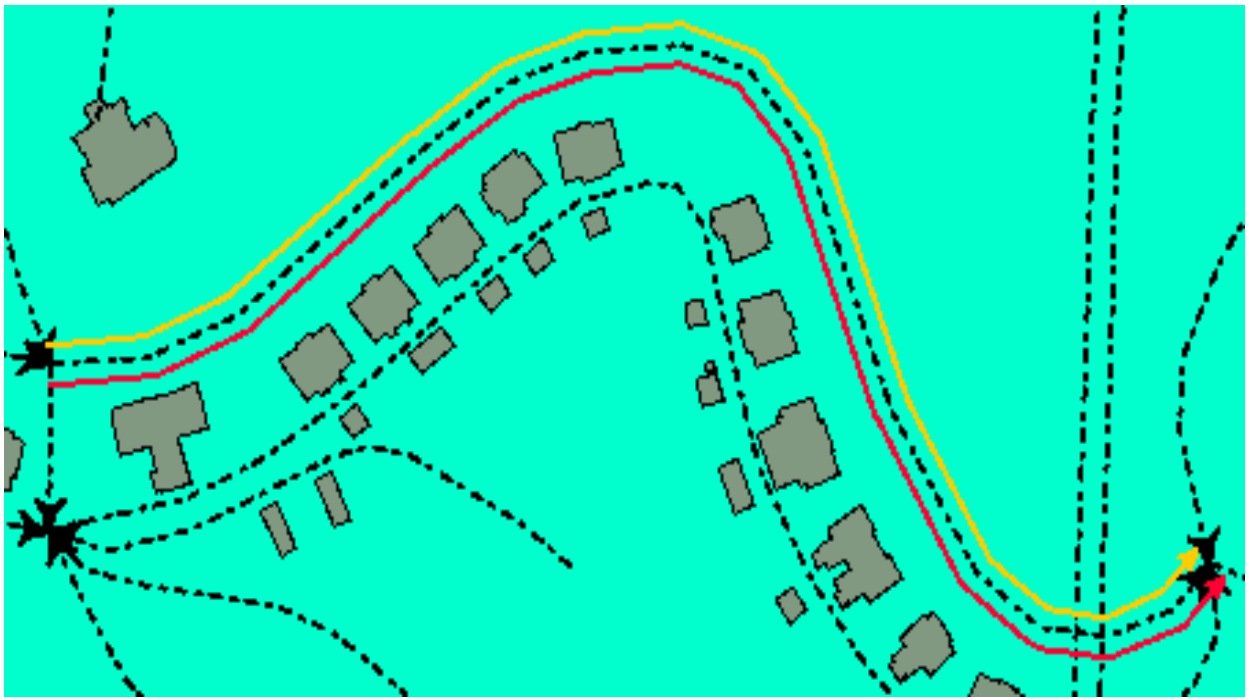
Vertex 6 (of A) can be snapped to Edge e7 (of B)

Detecting boundary gaps

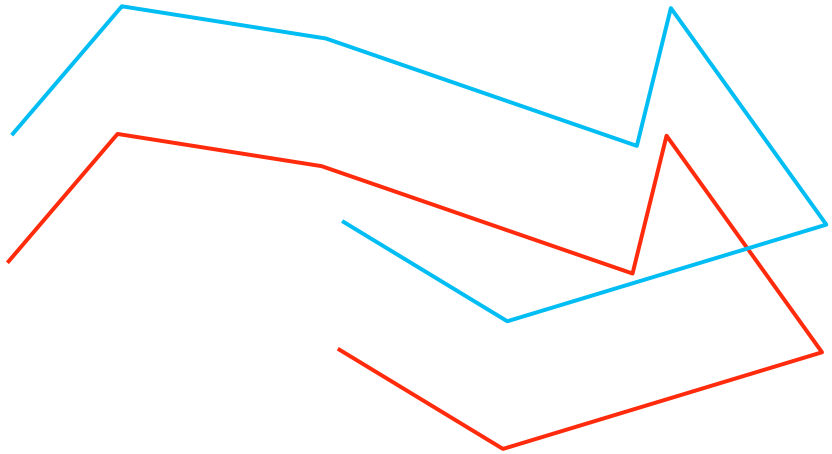
- Find the closest point on polygon B and replace the vertex from A with the closest point from B
  - Use SDO_UTIL.CLOSEST_POINTS function to find the closest point on e7 from vertex 6

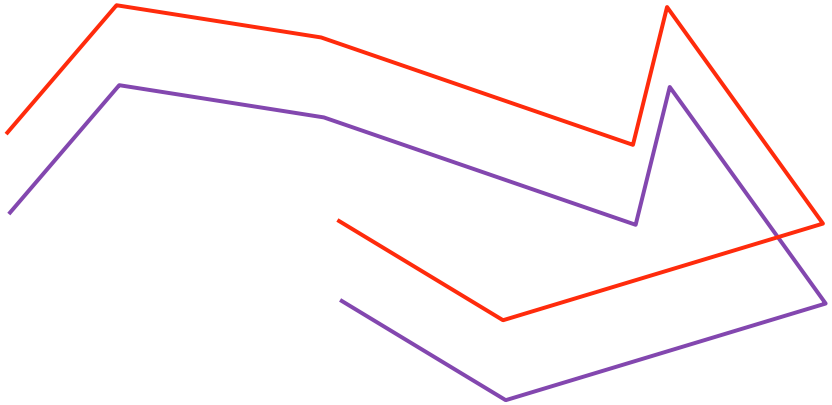


Generating parallel lines for road centerlines

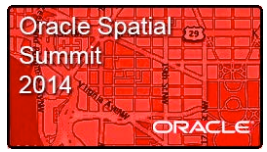


Will simple affine transform work ?



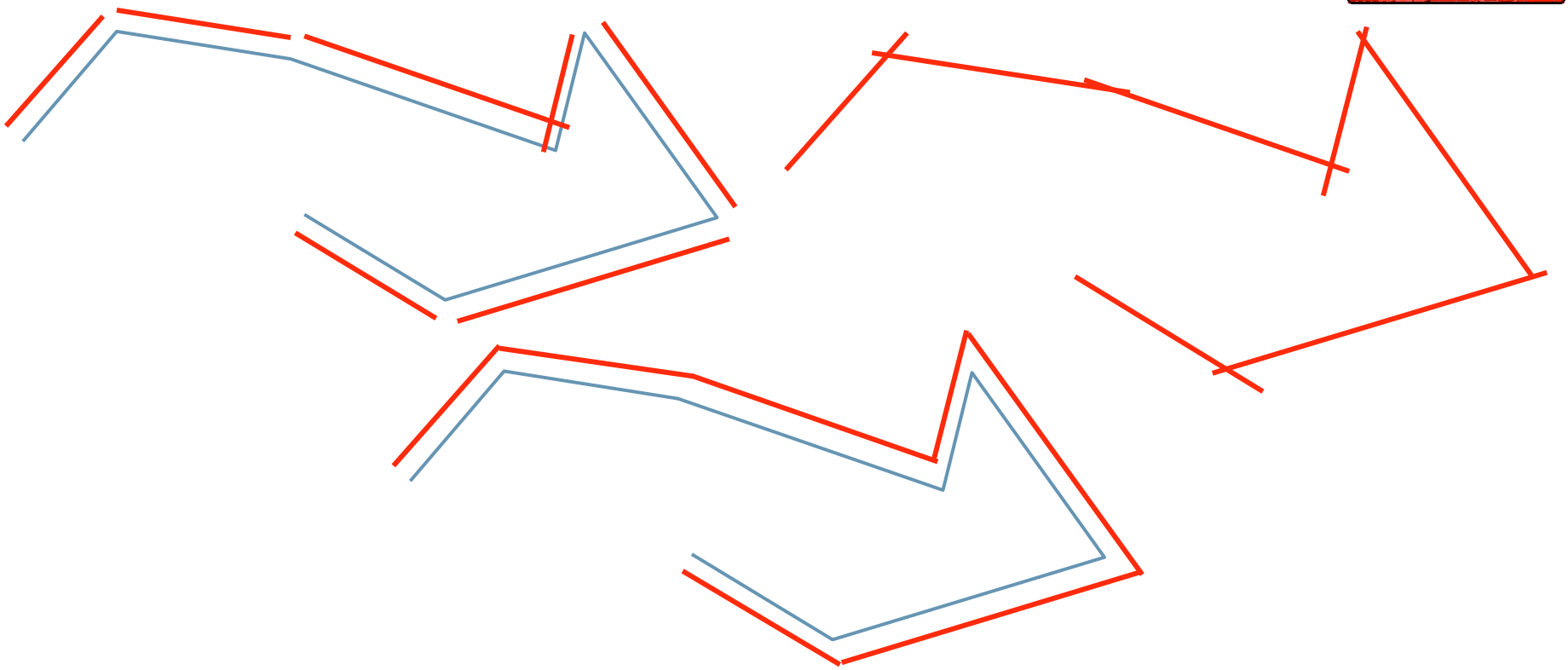


Examine each pair of coordinates



- Creating lines parallel to the original line requires working with edge of the line geometry
- Each edge is to be examined to compute the offset vector to use in the affine transform call
- Each translated edge is extended or shortened to ensure that the resulting lines edges are all touching
- Easy to incorporate unit parameter to such a function to create parallel lines at a specified distance

Creating parallel linear features



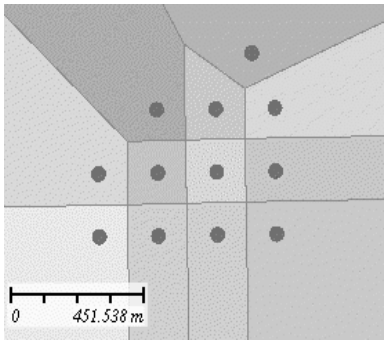
Using Spatial Java Stored Procedures

Open source Java code available for many functions

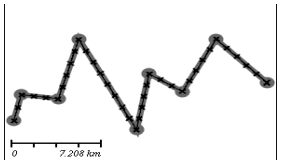
- JTS is one of the most popular java tool kits
- Easily integrates with Oracle Spatial functionality
- JTS defines the wrappers required to convert SDO_GEOMETRY to a JTS geometry class
- Makes it easy to expose any JTS supplied functions as PL/SQL methods in the database
- JTS is complementary rather than competitive

Some useful functions from JTS

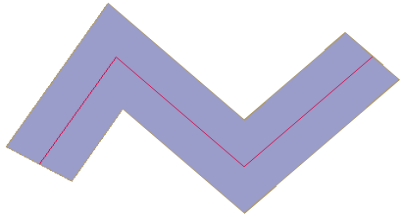
- Voronoi diagram generation



- Line densification



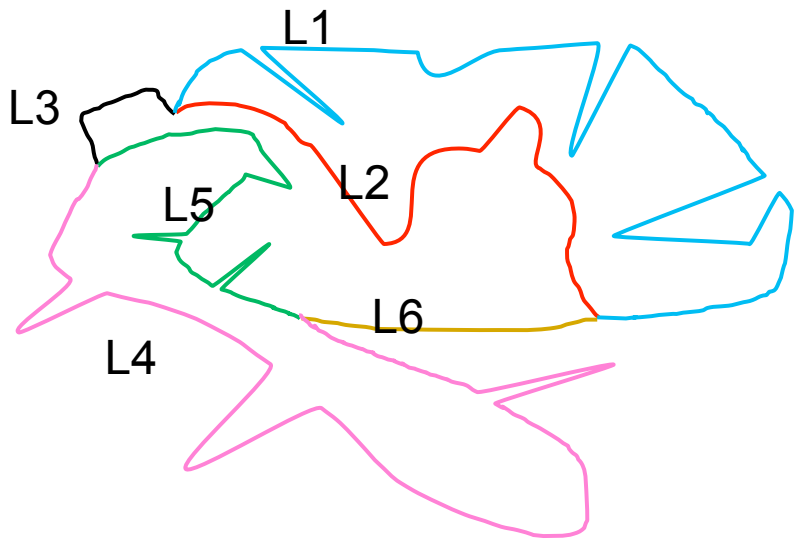
- Square Buffer



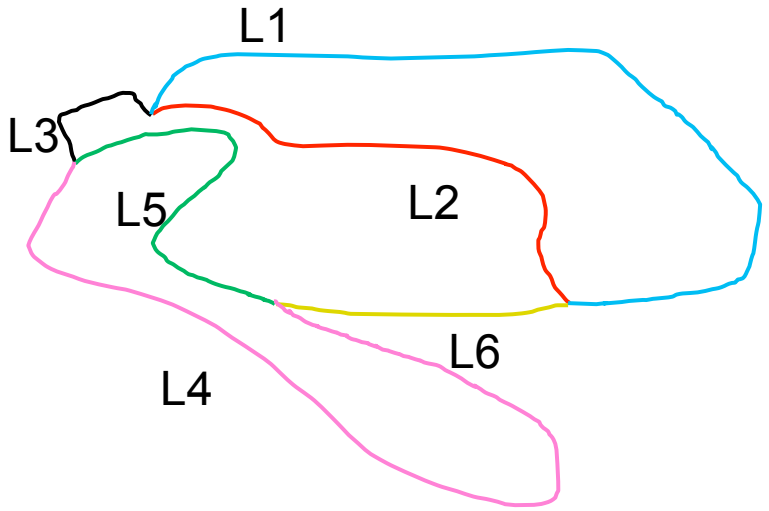
Topological Generalization of Features

- Producing cartographic maps at different scales requires generalization of features
- Object level generalization of individual features can generate a map that is not topologically consistent with the underlying data

Valid Simplification of Polygons

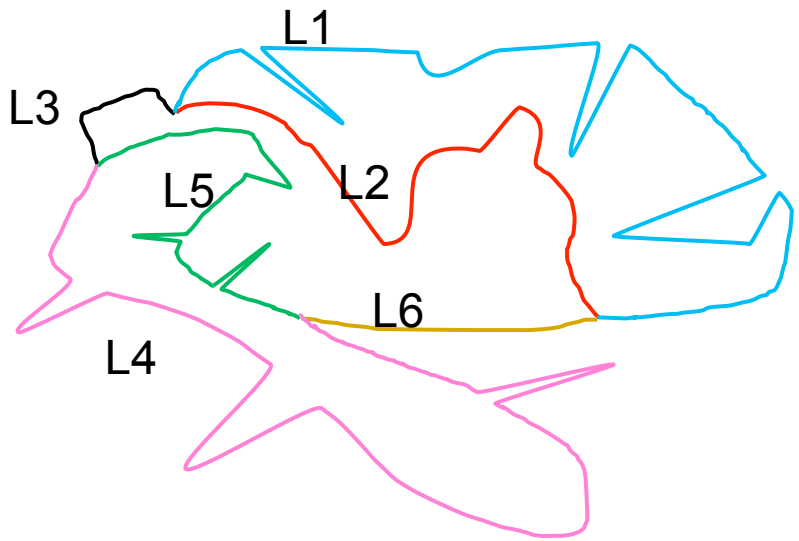


Input data with 6 lines

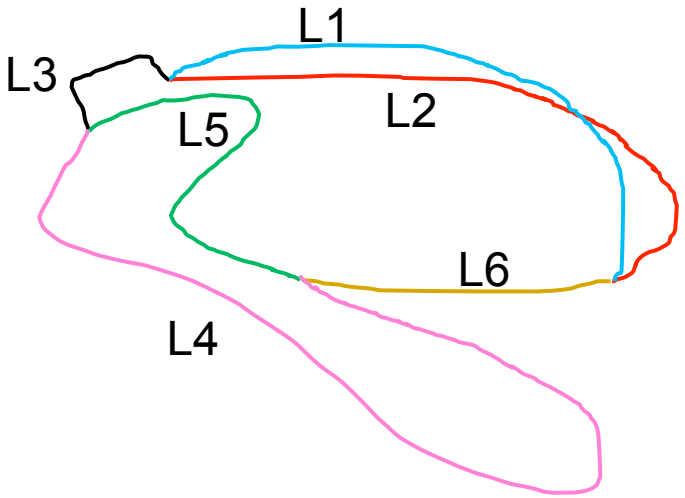


A valid simplification

Invalid Simplification

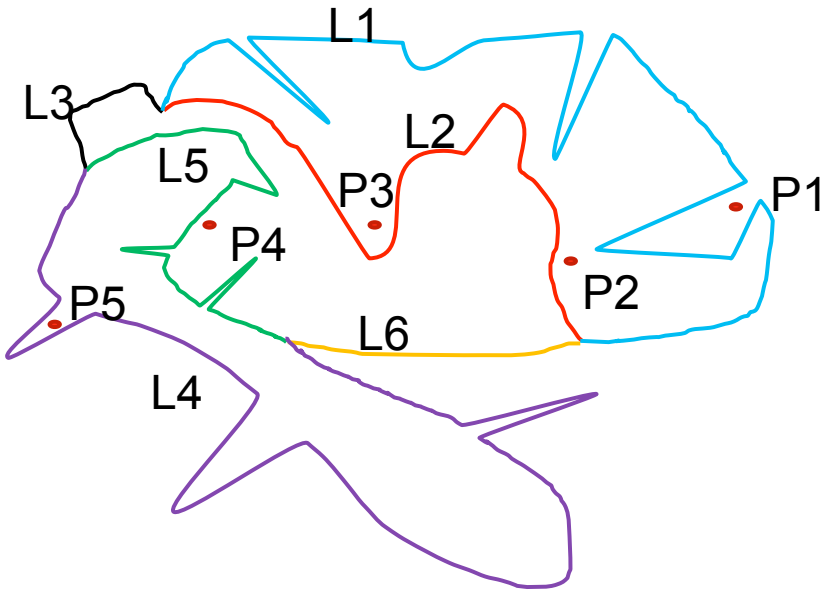


Input data with 6 lines

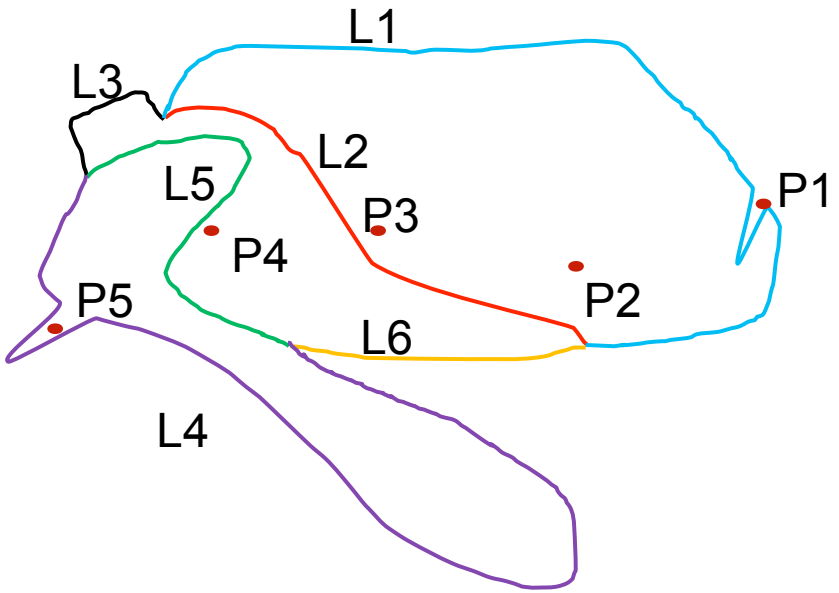


Invalid simplification

Valid Simplification of Polygons and Points

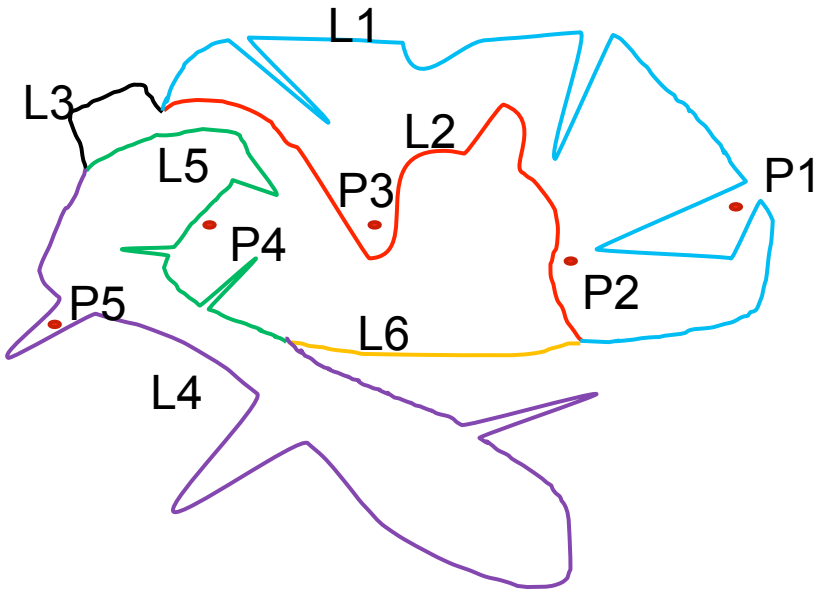


In put data with 6 lines and 5 point features

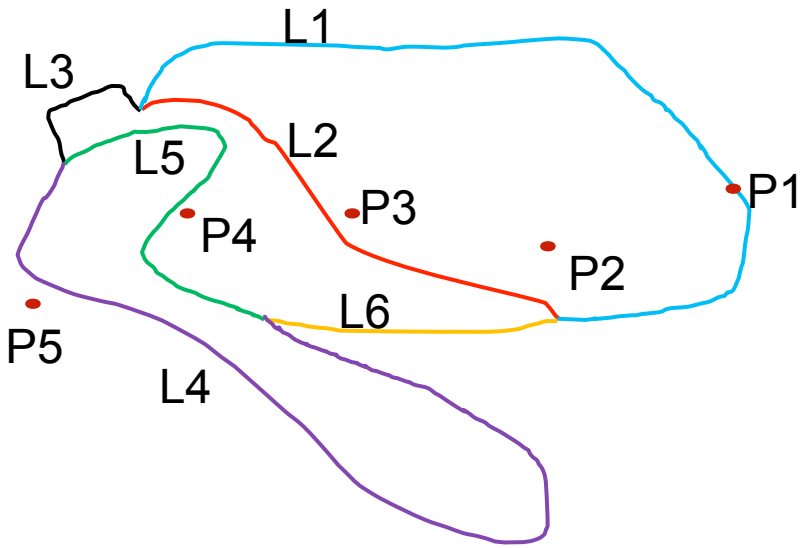


A valid simplification of the 6 lines with 5 point features

Invalid Simplification



In put data with 6 lines and 5 point features



Invalid simplification of the 6 lines 5 point features

Points P1 and P5 change their relative positions with respect to L1 and L4

Topology based generalization

- Construct a topology data model for all features in the map
- All polygonal, linear and point features are added to the topology
- Topology features are created corresponding to each of the original features
- Topology data model provides Get_Geometry function to construct the geometry for the feature based on topological elements
- Each edge of the topology is generalized to reduce the number of vertices
- Get_Geometry is used to reconstruct the feature geometry with reduced number of vertices

Topological Simplification

1. Simplify the edge geometry

→

2. Check if the topology is changing

NO ↓

3a. Commit the changes to the edge and go to step 1 for the next edge

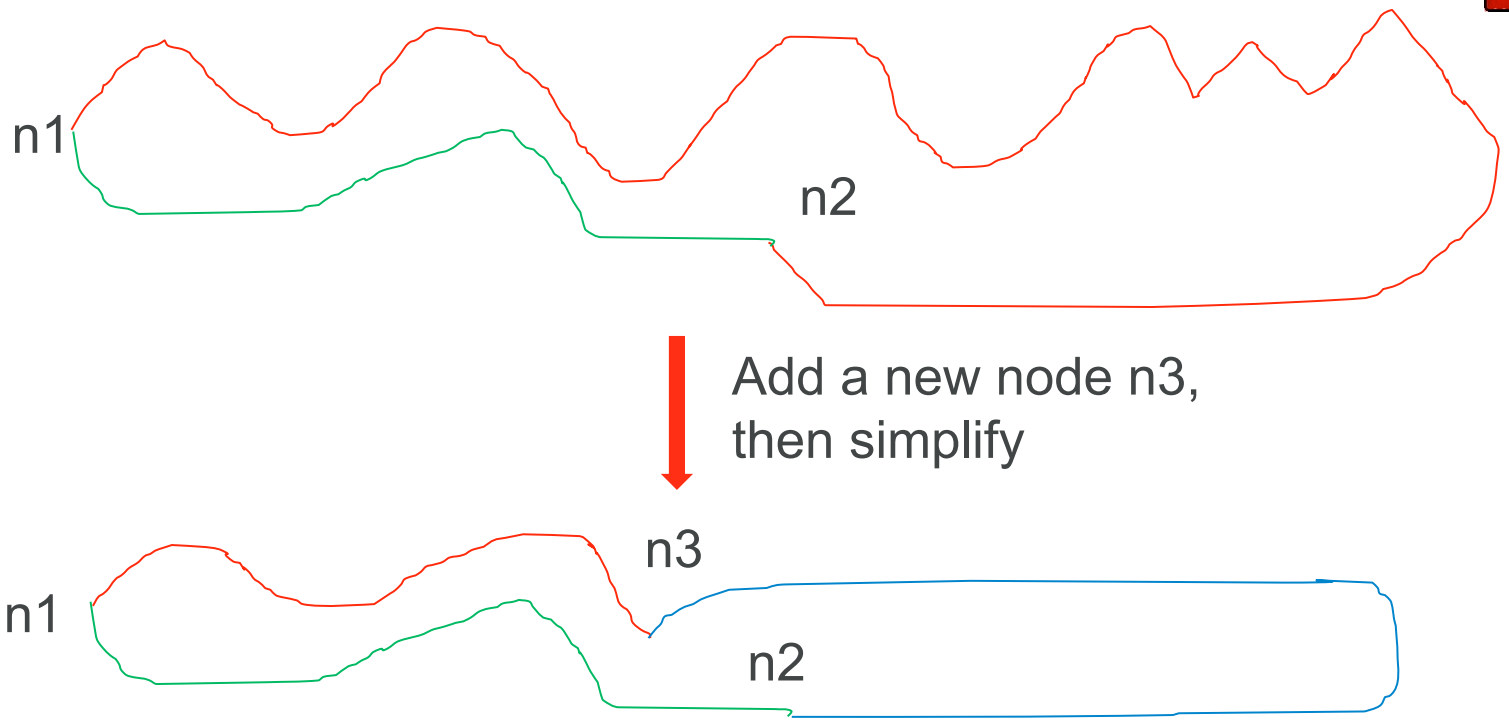
↓ YES

3b. Discard changes to the edge and go to step 1 for the next edge

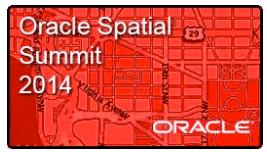
Simplifying Long Edges

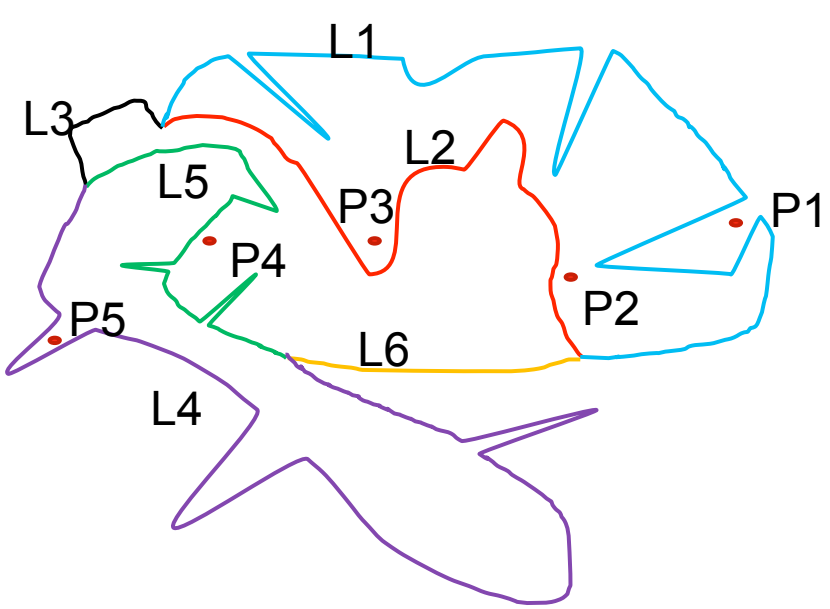
- In the default case, either a edge is completely simplified or not
  - In some cases, if an edge is very long, simplifying the whole edge may not be possible
  - But this leaves many vertices that can otherwise be removed
- Break these long edges by adding additional topological nodes
  - Break any non-simplified edges with vertices more than a threshold into more than one edge
  - Then simplify each of the smaller edges

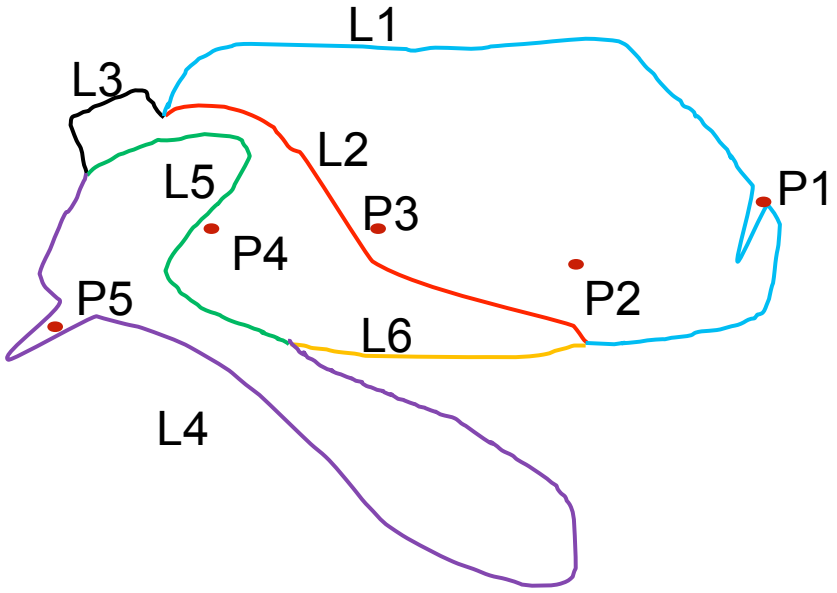
Adding nodes to long edges



Result of a Topology based Simplification



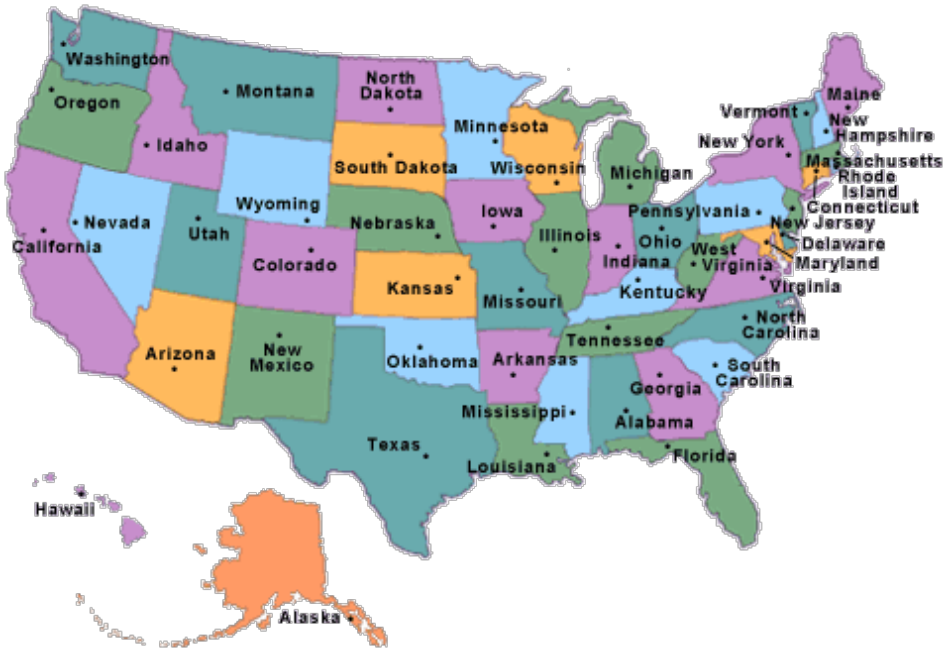




Use database views to share the same data for different use cases

- Geographic applications need the exact location, cartographic applications might need objects in different locations to create a pretty map
- Land management application and a report generation application share the same data
- But the report generation application wants the 48 states of US and Alaska and Hawaii next to each other on the same page
- Don’t duplicate the data to move Hawaii and Alaska to feed the reporting application
- Use database view with spatial functions

Cartographic Map



Create a view with Affine Transformations

- Create a function to transform Alaska and Hawaii

.. Function transform(state in varchar2)

… if (state='AK') then return sdo_util.affinetransform(geom,….);

elsif (state='HI') then return sdo_util.affinetransorm(geom,…)

else return geom;

- Create a view based on this function

Create view states_for_report as

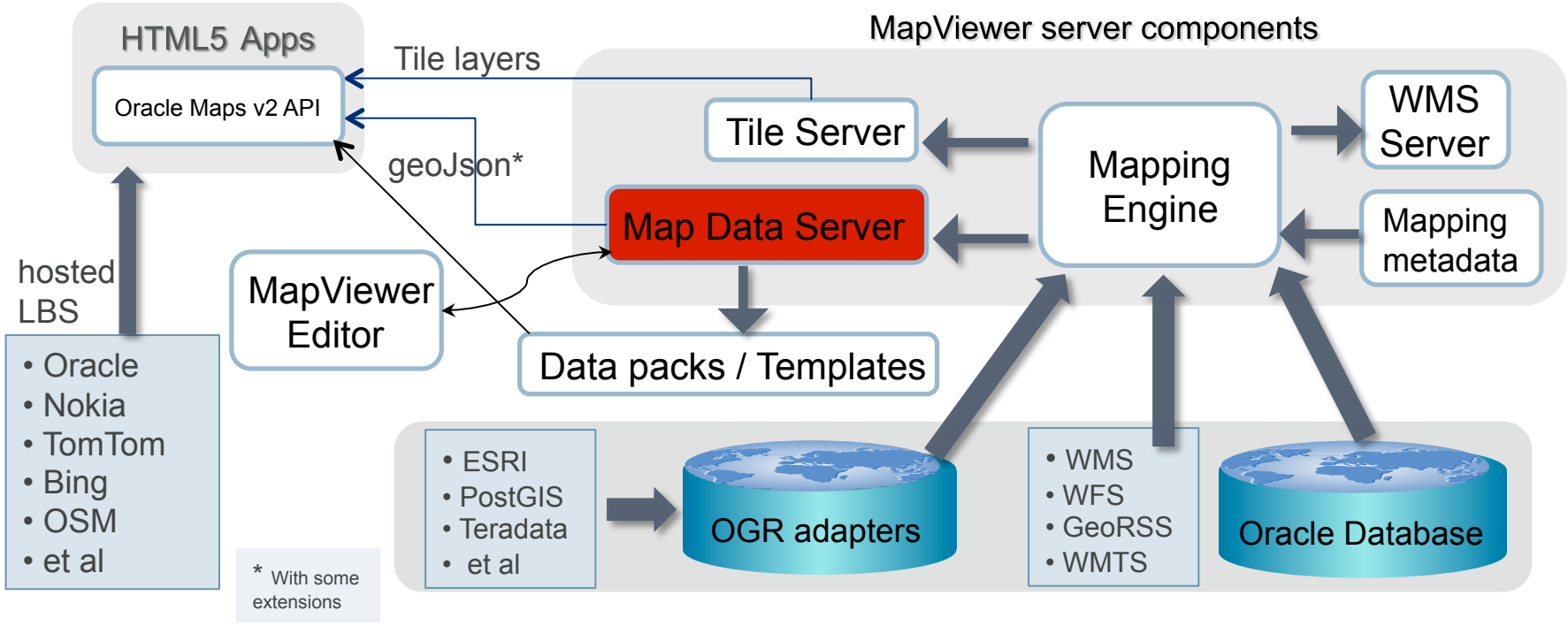
Select state, transform(geom), … from states

- The reporting application reads data from the STATES_FOR_REPORT view and the other application read data from STATES

Share data across different applications

- Use a data server to easily ship data via a web service
- Users can specify any SQL level constraints on tables to select relevant data
- MapViewer data server makes it easy to do this
- Publishing data as GeoJSON files

MapViewer Data Server



Data Server HTTP requests

- Sample MDS request to get states data, based on a predefined theme

/mapviewer/dataserver/mvdemo2?t=theme_demo_states&id_col=state_abrv&include_label_box=true

- Sample MDS request to get USA Zip Codes with simplification applied on the server side (12c only), based on a dynamic query :

/mapviewer/dataserver/mvdemo2?t=usa_zipcode&sql=select POSTALCODE, state, cnty, zippy, pc_name, pc_type, pa_name, geometry from zip_us&id_col=postalcode&include_label_box=true&simplify=true&threshold=95&dadp=4

Supported URL query parameters

- **t**: theme name (mandatory)
- **to_srid**: data should be tranformed into this SRID before sending to the client (optional)
- **dadp**: all cooridnates should have this number of digits after the decimal point; default is 5 (optional)
- **sql**: (only used when requesting dynamic query-based theme data) specifis a complete SQL query (mandatory)
- **paramnum**: specifies the number of bind variables (to be used for a pre-defined theme that has bind variables in its query condition) included in the request

What if we want to ship generalized data

- Starting with MapViewer 12.1.3
- **simplify**: indicates whether geometry should be simplified by the server
- **threshold**: if simplify is true, this value specifies the reduction percentage (value must be 1 through 99); e.g. threshold=75 means the geometry should be simplified by 75%, keeping only 25% of the vertices

Summary

- Use a good data model independent of application
- Use as many database features as possible
- Don’t reinvent the features that are already available in the database
- Use stored procedures for data management tasks
- MapViewer has many new powerful features to complement the Spatial features in the database

ORACLE

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

- **Next:**
  Lunch – Awards & SIG Presentation in Room 150AB
  (11:45am-1:15pm)

- **4:30-5:00pm: Meet the Experts – roundtable Q&A on topics in Room 150AB**
  - **Spatial Performance**
  - **Upgrading/Testing Apps for Spatial 12c**
  - **Raster & 3D**
  - **MapViewer/BI**
  - **Certification**
  - **SIG User Group**
- **Closing Reception (5:00) – Exhibit Hall**