Agenda

- Create database structures
- Load Spatial Data
- Index
- Issue SQL queries
- Develop simple Oracle Application Server Mapviewer application
What is a Spatial Database?

Spatial Data Types

Spatial Analysis

Spatial Indexing

All Spatial Data Stored in the Database

Fast Access to Spatial Data

Spatial Access Through SQL
Create Required Database Structures
All Spatial Types in Oracle 10g

- Networks (lines)
- Locations (points)
- Parcels (polygons)
- Imagery (raster, grids)
- Addresses (geocoded points)
- Topological Relations (persistent topology)
### Vector Map Data in Oracle Tables

#### Road

<table>
<thead>
<tr>
<th>ROAD_ID</th>
<th>NAME</th>
<th>SURFACE</th>
<th>LANES</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pine Cir.</td>
<td>Asphalt</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2nd St.</td>
<td>Asphalt</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3rd St.</td>
<td>Asphalt</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
The MDSYS Schema

• When Oracle Locator or Spatial is installed, the MDSYS user is created
  - Owner of Spatial types, packages, functions, procedures, metadata
  - Similar to user SYS
  - Privileged user
    - With ADMIN option

• This account is locked by default
  - Be careful with this administrative account
  - You should never need to log in as MDSYS
  - Never create any data as user MDSYS
SDO_GEOMETRY Object

- SDO_GEOMETRY Object

<table>
<thead>
<tr>
<th>SDO_GTYPE</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDO_SRID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SDO_POINT</td>
<td>SDO_POINT_TYPE</td>
</tr>
<tr>
<td>SDO_ELEM_INFO</td>
<td>SDO_ELEM_INFO_ARRAY</td>
</tr>
<tr>
<td>SDO_ORDINATES</td>
<td>SDO_ORDINATE_ARRAY</td>
</tr>
</tbody>
</table>

- Example

```sql
SQL> CREATE TABLE states (  
2     state       VARCHAR2(30),  
3     totpop      NUMBER(9),  
4     geom        SDO_GEOMETRY);  
```
SDO_GEOMETRY Object

- **SDO_POINT_TYPE**

  
<table>
<thead>
<tr>
<th>x</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>NUMBER</td>
</tr>
<tr>
<td>z</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

- **SDO_ELEM_INFO_INFO_ARRAY**

  VARRAY (1048576) OF NUMBER

- **SDO_ORDINATE_ARRAY**

  VARRAY (1048576) OF NUMBER
**SDO_GEOMETRY Object**

- **SDO_GTYPE** - Defines the type of geometry stored in the object

<table>
<thead>
<tr>
<th>GTYPE</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 POINT</td>
<td>Geometry contains one point</td>
</tr>
<tr>
<td>2 LINestring</td>
<td>Geometry contains one line string</td>
</tr>
<tr>
<td>3 POLYGON</td>
<td>Geometry contains one polygon</td>
</tr>
<tr>
<td>4 HETEROGENEOUS COLLECTION</td>
<td>Geometry is a collection of elements of different types: points, lines, polygons</td>
</tr>
<tr>
<td>5 MULTIPOINT</td>
<td>Geometry has multiple points</td>
</tr>
<tr>
<td>6 MULTILINESTRING</td>
<td>Geometry has multiple line strings</td>
</tr>
<tr>
<td>7 MULTIPOLYGON</td>
<td>Geometry has multiple polygons</td>
</tr>
</tbody>
</table>
### SDO_GTYPE

<table>
<thead>
<tr>
<th>SDO_GTYPE</th>
<th>Four digit GTYPEs - Include dimensionality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2D</td>
</tr>
<tr>
<td>1 POINT</td>
<td>2001</td>
</tr>
<tr>
<td>2 LINESTRING</td>
<td>2002</td>
</tr>
<tr>
<td>3 POLYGON</td>
<td>2003</td>
</tr>
<tr>
<td>4 COLLECTION</td>
<td>2004</td>
</tr>
<tr>
<td>5 MULTIPoint</td>
<td>2005</td>
</tr>
<tr>
<td>6 MULTILINESTRING</td>
<td>2006</td>
</tr>
<tr>
<td>7 MULTIPOLYGON</td>
<td>2007</td>
</tr>
</tbody>
</table>
Constructing Geometries

```sql
SQL> INSERT INTO LINES VALUES ( attribute_1, ..., attribute_n, 
    SDO_GEOMETRY ( 2002, null, null, 
    SDO_ELEM_INFO_ARRAY (1,2,1), 
    SDO_ORDINATE_ARRAY ( 10,10, 20,25, 30,10, 40,10) ) );
```

(20,25)  (10,10)  (30,10)  (40,10)
# How Spatial Data Is Stored

## Data type

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUST_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(40)</td>
</tr>
<tr>
<td>FIRST_NAME</td>
<td></td>
<td>VARCHAR2(20)</td>
</tr>
<tr>
<td>ADDR1</td>
<td></td>
<td>VARCHAR2(60)</td>
</tr>
<tr>
<td>ADDR2</td>
<td></td>
<td>VARCHAR2(40)</td>
</tr>
<tr>
<td>CITY</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>STATE</td>
<td></td>
<td>VARCHAR2(20)</td>
</tr>
<tr>
<td>ZIP</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>GEOLOC</td>
<td></td>
<td>NDSYS.SDO_GEOMETRY</td>
</tr>
<tr>
<td>PROPERTY_VALUE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>PROPERTY_DESCRIPTION</td>
<td></td>
<td>VARCHAR2(2000)</td>
</tr>
<tr>
<td>PROFITABILITY</td>
<td></td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

## Geographic coordinates

- **Liu**: SDO_GEOMETRY(1, NULL, SDO_POINT_TYPE=>'122.02415, 37.344318, NULL), NULL, NULL)
- **Crow**: SDO_GEOMETRY(1, NULL, SDO_POINT_TYPE=>'122.39411, 37.786136, NULL), NULL, NULL)
Spatial Metadata

- The spatial routines require you to populate a view that contains metadata about SDO_GEOMETRY columns.
- The metadata view is created for all Oracle Spatial users when Oracle Spatial is installed.
- The metadata view is called USER_SDO_GEOM_METADATA.
- For every SDO_GEOMETRY column, insert a row in the USER_SDO_GEOM_METADATA view.
SQL> DESCRIBE USER_SDO_GEOM_METADATA

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(32)</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(1024)</td>
</tr>
<tr>
<td>DIMINFO</td>
<td></td>
<td>SDO_DIM_ARRAY</td>
</tr>
<tr>
<td>SRID</td>
<td></td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

- **MDSYS.SDO_DIM_ARRAY**

  VARARRAY(4) OF SDO_DIM_ELEMENT

- **MDSYS.SDO_DIM_ELEMENT** object

<table>
<thead>
<tr>
<th>SDO_DIMNAME</th>
<th>VARCHAR2(64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDO_LB</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SDO_UB</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SDO_TOLERANCE</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>
Populating the USER_SDO_GEOM_METADATA View

SQL> INSERT INTO USER_SDO_GEOM_METADATA
2> (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
3> VALUES ('ROADS', 'GEOMETRY',
4> SDO_DIM_ARRAY (
5> SDO_DIM_ELEMENT('Long', -180, 180, 0.5),
6> SDO_DIM_ELEMENT('Lat', -90, 90, 0.5)),
7> 8307);

Note: For geodetic data, the x axis bounds must be –180 to 180, and y axis bounds –90 to 90.
Load Spatial Data into Oracle Spatial Database
Loading Spatial Data

- Categories of loading:
  - Bulk loading of data
    - SQL*Loader
    - Import
  - Transactional inserts
    - INSERT statement
  - Loading using Partner Tools
    - Example – SAFE Software’s FME
Validating Geometries

- Oracle Spatial validation routines ensure spatial data in Oracle Spatial is valid
  - SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT
    - Determines if a geometry is valid
  - SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT
    - Determines if all geometries in a layer are valid
- If data is invalid, both routines return why and where the geometry is invalid
FME Workbench
FME Mapping
Oracle Structures

SQL> describe streets

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT_CREATE</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>DT_RETIRE</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>ORG_CODE</td>
<td>NUMBER(11)</td>
<td></td>
</tr>
<tr>
<td>JOBSITE_ID</td>
<td>CHAR(10)</td>
<td></td>
</tr>
<tr>
<td>NUM</td>
<td>CHAR(12)</td>
<td></td>
</tr>
</tbody>
</table>
| PUB_AGG      | NUM  

SQL> select * from user_sdo_geo_data where table_name = 'STREETS';

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN_NAME</td>
<td></td>
</tr>
<tr>
<td>DIHINFO</td>
<td></td>
</tr>
<tr>
<td>SDO_DIM_ARRAY</td>
<td>(4, -180, 180, 0.01), SDO_DIM_ARRAY('Y', 90, 90, 5)</td>
</tr>
<tr>
<td>SDO_DIM_ARRAY</td>
<td>(X, -180, 180, 0.01)</td>
</tr>
<tr>
<td>SDO_DIM_ARRAY</td>
<td>(Y, -90, 90, 5)</td>
</tr>
<tr>
<td>SDO_DIM_ARRAY</td>
<td>(Z, 180, 180, 0.01)</td>
</tr>
</tbody>
</table>

SQL>
Set up Spatial Indexes
Spatial Indexing

- Used to optimize spatial query performance
- R-tree Indexing
  - Based on minimum bounding rectangles (MBRs) for 2D data or minimum bounding volumes (MBVs) for 3D data
    - Indexes two, three, or four dimensions
- Provides an exclusive and exhaustive coverage of spatial objects
- Indexes all elements within a geometry including points, lines, and polygons
Optimized Query Model

Layer Data

Primary Filter Spatial Index

Reduced Data Set

Secondary Filter Spatial Functions

Exact Result Set

Column where coordinates are stored

Index retrieves area of interest (window)

Procedures that determine exact relationship
A Look at R-tree Index Structures

create index GEOD_STATES_SIDX
    on GEOD_STATES (GEOM)
    indextype is MDSYS.Spatial_INDEX;

Index Information

Table MDRT_7B50$
Issue SQL Queries
Spatial Operators

- Full range of spatial operators
  - Implemented as functional extensions in SQL
  - Topological Operators
    - Inside
    - Contains
    - Touch
    - Disjoint
    - Covers
    - Covered By
    - Equal
    - Overlap Boundary
  - Distance Operators
    - Within Distance
    - Nearest Neighbor
Spatial Operators

- Operators
  - **SDO_FILTER**
    - Performs a primary filter only
  - **SDO_RELATE** and **SDO_<relationship>**
    - Performs a primary and secondary filter
  - **SDO_WITHIN_DISTANCE**
    - Generates a buffer around a geometry and performs a primary and optionally a secondary filter
  - **SDO_NN**
    - Returns nearest neighbors
**SDO_FILTER Example**

- Find all the cities in a selected rectangular area
- Result is approximate

```sql
SELECT c.city, c.pop90
FROM proj_cities c
WHERE sdo_filter (c.location,
                  sdo_geometry (2003, 32775, null,
                                sdo_elem_info_array (1,1003,3),
                                sdo_ordinate_array (1720300,1805461,
                                                    1831559, 2207250))
          ) = 'TRUE';
```

**Hint 1:** All Spatial operators return `TRUE` or `FALSE`. When writing spatial queries always test with `= 'TRUE'`, never `<> 'FALSE'` or `= 'true'`. 

Copyright © Oracle Corporation, 2006. All rights reserved.
SDO_RELATE Example

- Find all counties in the state of New Hampshire

```
SELECT c.county, c.state_abrv
FROM geod_counties c,
     geod_states s
WHERE s.state = 'New Hampshire'
     AND sdo_relate (c.geom,
                    s.geom,
                    'mask=INSIDE+COVEREDBY')
       = 'TRUE';
```

Note: For optimal performance, don’t forget to index GEOD_STATES(state)
Relationship Operators Example

• Find all the counties around Passaic county in New Jersey:

```sql
SELECT /*+ ordered */ a.county
FROM geod_counties b,
     geod_counties a
WHERE b.county = 'Passaic'
  AND b.state = 'New Jersey'
  AND SDO_TOUCH(a.geom,b.geom) = 'TRUE';
```

• Previously:

```sql
... AND SDO_RELATE(a.geom,b.geom,
    'MASK=TOUCH') = 'TRUE';
```
**SDO_WITHIN_DISTANCE Examples**

- Find all cities within a distance from an interstate

```sql
SELECT /*+ ordered */ c.city
FROM geod_interstates i, geod_cities c
WHERE i.highway = 'I170'
    AND sdo_within_distance (c.location, i.geom,
                              'distance=15 unit=mile') = 'TRUE';
```

- Find interstates within a distance from a city

```sql
SELECT /*+ ordered */ i.highway
FROM geod_cities c, geod_interstates i
WHERE c.city = 'Tampa'
    AND sdo_within_distance (i.geom, c.location,
                              'distance=15 unit=mile') = 'TRUE';
```
**SDO_NN Example**

- Find the five cities nearest to Interstate I170, ordered by distance

```sql
SELECT /*+ ordered */
    c.city, c.state_abrv,
    sdo_nn_distance (1) distance_in_miles
FROM geod_interstates i,
     geod_cities c
WHERE i.highway = 'I170'
    AND sdo_nn(c.location, i.geom,
                  'sdo_num_res=5 unit=mile', 1) = 'TRUE'
ORDER by distance_in_miles;
```

- Note: Make sure you have an index on `GEOD_INTERSTATES` (HIGHWAY).
Spatial Functions

- Returns a geometry
  - Union
  - Difference
  - Intersect
  - XOR
  - Buffer
  - CenterPoint
  - ConvexHull

- Returns a number
  - LENGTH
  - AREA
  - Distance
DEMO
SQL Developer
Develop Simple Oracle Application Server MapViewer Application
Oracle Spatial 10g Platform

Any Device

Partner Spatial Solution

Customer Built Solution

Oracle & Partner Business Applications

Middle Tier

3rd Party Tools
Wireless Midware
MapViewer
Oracle Application Server 10g

3rd Party Tools
LBS APIs
Web Services

Spatial Data Server

Spatial
Oracle10g

Oracle Location Technology

Oracle Core Technologies

Telemetry Services

Weather

Sensors

Field Obs.

SOAP, WSDL

Oracle Application Server 10g

Oracle Core Technologies

Oracle Location Technology

Oracle Spatial 10g Platform

Copyright © Oracle Corporation, 2006. All rights reserved.
MapViewer Overview

- A map rendering service in Oracle Application Server 10g. It is a server component (not a client viewer!)
- It visualizes data managed by Oracle Spatial.
- Provides a comprehensive set of APIs (XML and Java-based), using which client viewers can be easily developed and OGC WMS APIs
- Provides an enterprise-level solution to mapping metadata management.
MapViewer Overview

Architecture

Client

Browser/Application

Oracle Application Server 10g
(or standalone OC4J)

Mid-tier

MapViewer

Database

Oracle Spatial

Mapping metadata

HTTP

JDBC

Copyright © Oracle Corporation, 2006. All rights reserved.
MapViewer Query

A map request consists of:
- Base map name
- Center of map
- Width and height of map
- Optional tags
  - map name
  - jdbc_query
  - others

A map response consists of:
- A streamed map image or
- A URL to the map image along with the map MBR
MapViewer APIs

- MapViewer supports 3 API flavors
  - XML-based
    - Native language to MapViewer
  - Java thin library
    - a mapping “bean” (without UI)
  - JSP custom tags
    - a subset of functions
    - To be used as a ‘fast start’ for beginners
    - The JSP taglib can be easily added to Oracle JDeveloper’s component palette
    - A JDeveloper extension that lets you browse the current list of existing maps/themes/styles in a data source
Enhanced APIs and JDeveloper Integration
MapViewer Key Concepts

- Datasource
- Map
- Basemap
- Theme
- Style
MapViewer Welcome Page
http://localhost:8888/mapviewer

- Icon to go to/from the Admin page (see key icon in upper left)
- Several other hyperlinks, including Demos
On-line JSP Demos

These are the demos that come with your MapViewer deployment. Note that the last two demos require the demo dataset to be imported and a corresponding datasource defined. The demo dataset can be downloaded from CTN MapViewer site.

- view.jsp: A simple no hassle spatial data viewer
- mapclient.jsp: A simple JSP demo that uses a Java bean.
  This demo works with any Oracle Spatial dataset as long as you have defined a datasource for it.
- mapinit.jsp: A JSP demo using the MapViewer bean.
  This demo works only if you have imported the demo dataset and defined a datasource named "mvdemo".
- mapmap.jsp: A demo using MapViewer JSP tags and the bean.
  This demo works only if you have imported the demo dataset and defined a datasource named "mvdemo".
- New demos for 10.1.2: A set of demos for new features in MapViewer 10.1.2.
Oracle Map Builder

- Replacement for the Map Definition tool
- Currently in Beta and available on OTN for download
DEMO
Mapviewer and Mapbuilder
ORACLE IS THE INFORMATION COMPANY