Overview

• **Esri Integration at PennDOT**
  – History
  – Challenges
  – Solutions

• **Application of Lessons Learned**
  – Data Structure
  – Data Validation
  – Web Mapping (demo)
Esri Integration at PennDOT
History

• **Prior to 2012**
  – Heavily invested in Intergraph
  – Esri products used by analysts only
  – Datasets were exported to .shp files for consumption by Esri products
  – Oracle Spatial processing of ETL

• **2012 Initial Enterprise Integration**
  – Enterprise Geodatabase / ArcGIS Servers installed/configured
  – First Web Application based on Esri technology launched as an internet-available application for public consumption
  – Intergraph products still utilized in tandem
History (continued)

Today
- All internal/external web mapping applications powered by Esri
- Intergraph products used by analysts only
- Most data sets are now consumable by both Esri and Intergraph technologies
- 12 ArcGIS Server nodes providing services over 15 inter/intranet mapping applications
- ArcGIS Online utilized
- Mobile solutions using ArcGIS Collector
- Oracle Spatial Processing of ETL
Challenges

- **Esri geodatabase rules**
  - Restrictions on multiple geometry types per layer
  - Restrictions on geometry collections (SDO_GTYPE = n004)
  - Several datasets containing one or more restricted configurations

- **Handling of resulting point/line “aggregate layers”**

- **Multi-platform interoperability**
Solutions - Esri Geodatabase Rules

Single Layer with Multiple Geometry Types (points/lines) in different records

- Two Separate Tables
- Create Two Views on a Table Partitioned by SDO_GTYPE
### Solutions - Esri Geodatabase Rules

#### Two Separate Tables

<table>
<thead>
<tr>
<th>CREATE TABLE EVENT_POINTS</th>
<th>CREATE TABLE EVENT_LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td></td>
</tr>
<tr>
<td>OBJECTID NUMBER(38) NOT NULL,</td>
<td>OBJECTID NUMBER(38) NOT NULL,</td>
</tr>
<tr>
<td>EVENTID NUMBER(38) NOT NULL,</td>
<td>EVENTID NUMBER(38) NOT NULL,</td>
</tr>
<tr>
<td>GEOMETRY MDSYS.SDO_GEOMETRY</td>
<td>GEOMETRY MDSYS.SDO_GEOMETRY</td>
</tr>
<tr>
<td>);</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CREATE INDEX EVENT_POINTS_SIDX</th>
<th>CREATE INDEX EVENT_LINES_SIDX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON EVENT_POINTS(GEOMETRY)</td>
<td>ON EVENT_LINES(GEOMETRY)</td>
</tr>
<tr>
<td>INDEXTYPE IS MDSYS.Spatial_INDEX</td>
<td>INDEXTYPE IS MDSYS.Spatial_INDEX</td>
</tr>
<tr>
<td>PARAMETERS (‘LAYER_GTYPE=POINT’);</td>
<td>PARAMETERS (‘LAYER_GTYPE=LINE’);</td>
</tr>
</tbody>
</table>
Solutions - Esri Geodatabase Rules

Two Separate Tables

• **Pros**
  - Easy set-up
  - Spatial index constrains geometry types able to be inserted

• **Cons**
  - Have to maintain each table independently
  - No easy way to handle geometry collections (SDO_GTYPE = n004)
  - Non-spatial queries will need to union both tables
Partition Single Table and Create Views

```sql
CREATE TABLE EVENT_POINTS
(
    OBJECTID NUMBER(38) NOT NULL,
    EVENTID NUMBER(38) NOT NULL,
    GEOMETRY MDSYS.SDO_GEOMETRY
)
PARTITION BY LIST(GEOMETRY.SDO_GTYPE)
(
    PARTITION P0 VALUES ( NULL,2004),
    PARTITION P1 VALUES (2001,2005),
    PARTITION P2 VALUES (2002,2006),
    PARTITION P3 VALUES (2003,2007),
)
ENABLE ROW MOVEMENT;

CREATE INDEX EVENT_POINTS_SIDX ON EVENT_POINTS(GEOMETRY) INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```
Partition Single Table and Create Views

CREATE VIEW EVENT_POINTS_V AS
SELECT
  OBJECTID,
  EVENTID,
  GEOMETRY
FROM EVENT_POINTS PARTITION(P1);

CREATE VIEW EVENT_LINES_V AS
SELECT
  OBJECTID,
  EVENTID,
  GEOMETRY
FROM EVENT_LINES PARTITION(P2);
Partition Single Table and Create Views

• **Pros**
  - Single table to maintain
  - Potential to use local partitioned
  - Partition pruning reduces total I/O to satisfy request
  - Sequesters n004 and NULL geometries

• **Cons**
  - Slightly more complex set up
  - Index GTYPE parameter can not be set
Solutions - Esri Geodatabase Rules

• **Single Record with Multiple Geometry Types (points/lines)**
  – Single table of attributes / linear geometry, table of points, view that combines point geometry and attributes
Solutions - Esri Geodatabase Rules

Single Attribute Table, Point Geometry Table, Point Attribute View

CREATE TABLE EVENT_LINES
(
    OBJECTID NUMBER(38) NOT NULL,
    EVENTID NUMBER(38) NOT NULL,
    ATTRIBUTE_1 VARCHAR2(50),
    ATTRIBUTE_2 VARCHAR2(20),
    GEOM_LENGTH NUMBER(8) NOT NULL,
    GEOMETRY MDSYS.SDO_GEOMETRY
);  
CREATE INDEX EVENT_LINES_SIDX
ON EVENT_LINES(GEOMETRY)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS ('LAYER_GTYPE=LINE');

CREATE TABLE EVENT_POINTS
(
    OBJECTID NUMBER(38) NOT NULL,
    L_OBJECTID NUMBER(38) NOT NULL,
    GEOMETRY MDSYS.SDO_GEOMETRY
);  
CREATE INDEX EVENT_POINTS_SIDX
ON EVENT_LINES(GEOMETRY)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS ('LAYER_GTYPE=POINT');
CREATE VIEW EVENT_POINTS_V AS
SELECT
  P.OBJECTID,
  L.EVENTID,
  L.ATTRIBUTE_1,
  L.ATTRIBUTE_2,
  L.GEOM_LENGTH,
  P.GEOMETRY
FROM EVENT_POINTS P
JOIN EVENT_LINES L ON
  P.L_OBJECTID = L.OBJECTID;
Solutions - Esri Geodatabase Rules

**Single Attribute Table, Point Geometry Table, Point Attribute View**

Ex: Draw points when geometry length is less/equal than 500 feet

```
SELECT
  OBJECTID,
  EVENTID,
  ATTRIBUTE_1,
  ATTRIBUTE_2,
  GEOM_LENGTH,
  GEOMETRY
FROM EVENT_POINTS_V
WHERE GEOM_LENGTH <= 500;
```

Ex: Draw lines when geometry length is greater than 500 feet

```
SELECT
  OBJECTID,
  EVENTID,
  ATTRIBUTE_1,
  ATTRIBUTE_2,
  GEOM_LENGTH,
  GEOMETRY
FROM EVENT_LINES
WHERE GEOM_LENGTH > 500;
```
Solutions - Esri Geodatabase Rules

Zoomed Out – Point (geom_length = 105)

Zoomed In – Line (geom_length = 105)
Single Attribute Table, Point Geometry Table, Point Attribute View

- **Pros**
  - Single set of attributes to maintain
  - Flexible display capabilities
  - Trigger updates point table

- **Cons**
  - More complex set up
  - Multiple tables to maintain
  - Requires scale dependent queries to display
Solutions – Handling of “Aggregate” Layers
Application of Lessons Learned
Application of Lessons Learned – Data Structure

Helpful Tips

• **Index GTYPE Parameter**
  – Use when possible – prevents geometries of the wrong type from being inserted

• **Data Projection**
  – Use a single projection when possible
  – On the fly re-projection is a performance hit

• **Partition Large Spatial Tables**
  – Local Partitioned Spatial Indexes
  – Carefully consider how they will be queried
Application of Lessons Learned – Data Structure

Helpful Tips (continued)

• **Simplify Geometry**
  – Reduce number of vertices to the scale at which you will be displaying data
  – `SDO_UTIL.SIMPLIFY`
    • Removes extraneous vertices within a given threshold
    • Based on Douglas-Peucker algorithm

• **Aggregate Geometry**
  – Group geometries where attributes match to reduce number of records
  – `SDO_AGGR_CONCAT_LINES`

• **Esri GeoProcessing Tools**
  – Sometimes faster than Oracle Spatial
Application of Lessons Learned – Data Validation

Helpful Tips (continued)

• Validate Geometry
  – `SDO_GEOM.VALIDATE_Layer_with_CONTEXT`
    • Validates the geometry column for an entire table
  – `SDO_GEOM.VALIDATE_Geometry_with_CONTEXT`
    • Validates the geometry column of a single record
Application of Lessons Learned – Data Validation

Helpful Tips (continued)

- **Validate Geometry** (continued)
  - `SDO_GEOM_UTIL.RECTIFY_GEOMETRY`
    - Fixes line/polygon geometry errors
      - Duplicate Vertices
      - Self intersecting polygons
      - Incorrect orientation of interior/exterior rings of a polygon
  - Does not modify valid geometries
  - Does not fix all issues (error returned if not able to fix)
Application of Lessons Learned (demo)
Points and Lines In Same Layer
Points and Lines In Same Geometry
Partitioned Table / Views
Aggregated Geometry
Special Thanks

• Jeff Fennimore – Information Technology Manager (PennDOT)
• Jianyang Shi – Database Administrator (PennDOT)
• George Nace – Database Administrator (GeoDecisions)
Questions?

Matthew J. Allen
Database Administrator
mallen@geodecisions.com