Working with the geodatabase effectively using SQL

Shannon Shields
Thomas Brown
Kevin Watt
Today’s schedule

- 75 minute session
  - 60 – 65 minute lecture
  - 10 – 15 minutes Q & A following the lecture
- Cell phones and pagers

Please!

*Turn OFF cell phones and paging devices*

- Please complete the *session survey* – we take your feedback very seriously!
Why Use a Spatial Type?

**Efficiency**
- Spatial data and methods are stored in the database
- Spatial operations are encapsulated within the type
- Applications access native dbms type – no mapping layer

**Accessed using common API’s and SQL**
- C, C++, C#, Java, OLEDB
- SQL extensions enable spatial types and methods in DML and select statement syntax
What is a Spatial Type

DBMS datatype and properties
- Geometry type, coordinates, dimension, spatial reference

Spatial Index
- Access path for quick retrieval

Operators and Functions
- Constructors
- Accessor
- Relational
- Geometry
Geodatabase geometry storage options

**SQL Server**
- SDEBinary
- SQL Server 2008 GEOMETRY
- GEOGRAPHY

**Oracle**
- SDEBinary
- SDELOB
- ST_GEOMETRY
- SDO_GEOMETRY

**DB2/Informix**
- ST_GEOMETRY

**PostgreSQL**
- ST_GEOMETRY
- PostGIS
- GEOMETRY
st_geometry Properties

- Geometry Type
- Extent (MBR)
- Characteristics
  - Simple, Empty, Dimension
- Spatial Reference
- Coordinates

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTITY</td>
<td>NUMBER (38)</td>
</tr>
<tr>
<td>NUMPTS</td>
<td>NUMBER (38)</td>
</tr>
<tr>
<td>MINX</td>
<td>FLOAT (64)</td>
</tr>
<tr>
<td>MINY</td>
<td>FLOAT (64)</td>
</tr>
<tr>
<td>MAXX</td>
<td>FLOAT (64)</td>
</tr>
<tr>
<td>MAXY</td>
<td>FLOAT (64)</td>
</tr>
<tr>
<td>AREA</td>
<td>FLOAT (64)</td>
</tr>
<tr>
<td>LEN</td>
<td>FLOAT (64)</td>
</tr>
<tr>
<td>SRID</td>
<td>NUMBER (38)</td>
</tr>
<tr>
<td>POINTS</td>
<td>BLOB</td>
</tr>
</tbody>
</table>
Spatial Index

Rtree or Grid, Grid-Tessellating

Modeled as separate table or DBMS index

Associated to geometry type and operators

Enabled to the optimizer using statistics
Operators and Functions

Constructors - Builds a new instance of the type
– Can be overloaded

\texttt{ST\_POINT} (X, Y, SRID)
\texttt{ST\_POINT} (X, Y, Z, M, SRID)
\texttt{ST\_POINT} ("X Y", SRID)

– Derived Subtypes can construct Supertype

\texttt{CREATE TABLE accidents (shape \texttt{ST\_GEOMETRY});}
\texttt{INSERT INTO accidents (\texttt{ST\_POINT}(10,10,1));}
A geometry’s internal representation is different from an applications external format to improve performance and storage efficiency.

Accessor functions convert geometry data stored in internal format to an external application format.
Operators and Functions

**Accessor**

– Well-Known Text

```sql
INSERT INTO districts VALUES (ST_POLYGON('polygon((10 10,50 10,50 50,10 50,10 10))',1));

SELECT ST_ASTEXT(A.SHAPE) FROM districts;
ST_ASTEXT(SHAPE)
-------------------------------------
POLYGON ((10.0 10.0, 50.0 10.0, 50.0 50.0, 10.0 50.0, 10.0 10.0))
```
Operators and Functions

Accessor
  – Well-Known Binary

```
SELECT ST_ASTEXT(ST_GEOMFROMWKB(ST_ASBINARY(SHAPE),1))
FROM districts

ST_ASTEXT(ST_GEOMFROMWKB(ST_ASBINARY(SHAPE),0))
---------------------------------------
POLYGON ((10.0 10.0, 50.0 10.0, 50.0 50.0, 10.0 50.0, 10.0 10.0))
```
• Tests the spatial relationship between two geometry objects
  – Input \((\text{GEOMETRY~A}, \text{GEOMETRY~B})\)
  – Returns TRUE / 1 or FALSE / 0

• Examples
  – \text{ST~INTERSECTS}
  – \text{ST~TOUCHES}
  – \text{ST~CONTAINS}
Relational operators

- $\text{ST_TOUCHES (Geometry}_A, \text{Geometry}_B) = 0 \text{ or } 1$
Geometry operators

• Builds a new geometry object from one or more existing geometry objects
  – Input: \((\text{GEOMETRY\_A, args...})\)
  – Returns: \((\text{GEOMETRY})\)

• Examples
  – \text{ST\_BUFFER}
  – \text{ST\_UNION}
  – \text{ST\_DIFFERENCE}
Geometry operators

- **ST_Union**
  
  \( \text{geom1, geom2} \)

- **ST_Difference**
  
  \( \text{geom1, geom2} \)
Using SQL with the Geodatabase
Editing ArcGIS feature classes with SQL

- **Edit simple features only**
  - Points, lines, polygons (single or multipart)
  - Without geodatabase behavior
    - Not part of topology, geometric network

- **Editing non-versioned feature classes**
  - Applies directly to business table (no delta tables)

- **Editing versioned feature classes requires a defined workflow**
Editing non-versioned feature classes

- Requires a unique identifier (objectid) when inserting
  - Obtained from classes sequence or procedure

- Can leverage DBMS functionality
  - Unique indexes, constraints, referential integrity, default values, triggers
Obtaining unique row_id values

• DBMS procedures for obtaining unique identifiers

//Oracle

SQL> SELECT registration_id FROM sde.table_registry
    WHERE owner = 'TOMB' AND table_name = 'PARCELS';

SQL> SELECT sde.version_user_ddl.next_row_id('TOMB', 114)
FROM dual;

//SQL*Server

SELECT registration_id FROM sde.sde_table_registry
WHERE owner = 'TOMB' AND table_name = 'PARCELS'

DECLARE @id AS INTEGER
DECLARE @num_ids AS INTEGER
exec sde.i114_get_ids 2, 1, @id OUTPUT, @num_ids OUTPUT
Editing Versioned feature classes

• Use SQL to access multiversioned views
  – Ability to modify geometry when stored as a spatial type
  – Documentation is located within ArcGIS Desktop Help

• Recommended workflow
  – Create multiversioned views
  – Create a new version
  – Perform edits within the new version
  – Use ArcMap to reconcile/post to its parent version
Working with multiversioned views

• For non-ESRI applications which require SQL access to versioned tables
  – Ability to access any version
  – View derives a result set based on a version query
  – Procedures provided with SDE installation

• SDE administration command for creating the view

  sdetable -o create_mv_view -T <view_name> -t <table_name>
  [-i <service>] [-s <server_name>] [-D <database>]
  -u <DB_User_name> [-p <DB_User_password>] [-N] [-q]

  sdetable -o create_mv_view -T parcels_mv -t parcels -i 5151
  -s alex -u tomb -N
Working with multiversioned views

- DBMS procedure for setting the version for the view to reference

//Oracle
SQL> exec sde.version_util.set_current_version ('tomb.PROPOSED_SUBDIVISION');

SQL> SELECT owner, parcel_id FROM parcel_mv
    WHERE st_envintersects(shape, 5,5,10,10) = 1;

//SQL*Server
exec sde.set_current_version ('tomb.PROPOSED_SUBDIVISION')
or
exec dbo.set_current_version ('tomb.PROPOSED_SUBDIVISION')

//DB2
call setcurrentversion ('tomb.PROPOSED_SUBDIVISION')
Working with multiversioned views

- DBMS procedures for editing a versioned geodatabase and multiversioned views

//Oracle
SQL> exec sde.version_user_ddl.edit_version ('tomb.PROPOSED_SUBDIVISION', 1);
SQL> UPDATE parcel_mv SET owner = 'Ethan Thomas'
WHERE parcel_id = '322-2002-001' AND st_intersects(shape, st_geom) = 1;
SQL> COMMIT;

SQL> exec sde.version_user_ddl.edit_version ('tomb.PROPOSED_SUBDIVISION', 2);

//SQL*Server
exec sde.edit_version ('tomb.PROPOSED_SUBDIVISION', 1)
exec dbo.edit_version ('tomb.PROPOSED_SUBDIVISION', 2)
Rules when working with multiversioned views

• Do not update the objectid (row_id) value

• Do not edit archive enabled classes in the DEFAULT version
  – See Knowledge Base article: 35645

• Do not modify geometries for classes participating in topologies or geometric networks
  – Will not create dirty areas or be validated
  – Will not maintain connectivity in the logical network

• Do not update attributes which define geodatabase behavior
  – Enabled/Disabled attributes
  – Ancillary attributes
  – Weight attributes
  – Subtypes
Spatial views

- Spatial views are stored as any other view -- in the database
- Views can be created with *sdetable* or created in the database and registered with ArcGIS
Registration

- Registration informs the geodatabase of tables or views containing spatial attributes created outside ArcGIS.

- Registration writes information about these tables and the spatial attribute into the geodatabase system tables so ArcGIS can access them properly:
  - Description of the table’s attributes
  - Type of features in the spatial attribute
  - Spatial reference information
Using ArcObjects

• Nothing prevents the application developer from consuming relational operators via ArcObjects

```csharp
//Oracle, st_geometry
pQueryDef = pFeatureWorkspace.CreateQueryDef();
pQueryDef.Tables = "tb.parcels", "tb.neighborhoods";
pQueryDef.SubFields = "parcels.objectid";
pQueryDef.WhereClause =
    "sde.st_intersects(parcels.shape, neighborhoods.shape) = 1";
pCursor = pQueryDef.Evaluate();
IRow pRow = pCursor.NextRow();
int pCnt = 0;
if (pRow != null)
{
    pCnt = pCnt + 1;
    Marshal.ReleaseComObject(pRow);
}
```
Working with PostgreSQL Spatial Types
What is PostgreSQL?

- Open Source RDBMS
- Conforms to SQL 92/99 standards
- Includes complex database features
  (Inheritance, stored procedures, UDT, views, extensible index framework, etc.)

ArcGIS Server Enterprise 9.3 includes

- ArcSde for PostgreSQL (installs PostgreSQL rdbms)
- Support for SQL Spatial types
Spatial types in PostgreSQL

- Two spatial types
  - `ST_GEOMETRY` (ESRI)
  - `GEOMETRY` (Open source project - PostGIS)

- Both are OGC/ISO compliant
  - Support standard constructor, accessor, analytical functions

- Full geodatabase functionality supported on both spatial types
  - Geometric networks, topology, versioning, archiving, replication etc.

- Both types provide spatial index functionality
What is different between the two spatial types?

**ST_GEOMETRY**
- Resides under ‘sde’ schema
- Consistent implementation across databases (Oracle, Informix, DB2, PostgreSQL)
- Supports parametric curves, surfaces, and point-id
- Stored as compressed shape

**GEOMETRY**
- Resides under ‘public’ schema
- Only available in PostgreSQL
- Not supported
- Stored as Well Known Binary
**st_geometry**

- Default geodatabase spatial type
  - GEOMETRY_STORAGE parameter value defines the storage type
    - ‘ST_GEOMETRY’ or ‘PG_GEOMETRY’
- Superclass `st_geometry` implemented as a UDT
- Subtype `(st_point, st_linestring, st_polygon, st_multipoint...)` created as a domain type

```sql
CREATE DOMAIN sde.st_point AS sde.st_geometry
  CONSTRAINT st_point_check
  CHECK ((point_constraint(VALUE) = true));
```

- Spatial Index
  - Uses GiST indexing framework
  - Implements Rtree indexing strategy
Creating data with st_geometry

\-- Create table with st_geometry column
CREATE TABLE gis.blocks_st (objectid INTEGER NOT NULL, block VARCHAR(24), shape st_geometry);

\-- Assign ownership
alter table gis.blocks_st owner to gis;

\-- Create spatial index
CREATE INDEX blockssp_idx ON gis.blocks_st USING gist(shape st_geometry_ops);

\-- Register with SRID
SELECT
sde.st_register_spatial_column('mydb','gis','blocks_st', 'shape',1);

\-- Insert Data
INSERT INTO gis.blocks_st VALUES (1,'block',
st_geometry('polygon((52 28,58 28,58 23,52 23,52 28))'1));
Creating data with Geometry

```
\\ Create table
CREATE TABLE gis.blocks_pg (objectidINTEGER NOT NULL, block VARCHAR(24));

\\ Add spatial column
SELECT public.Addgeographycolumn ('gis', 'blocks_pg', 'geom', -1, 'POLYGON', 2);

\\ Assign ownership
alter table gis.blocks_pg owner to gis

\\ Create spatial index
CREATE INDEX blockssp_idx ON blocks_st USING gist(geom
gist_geometry_ops

\\ Insert data
INSERT INTO gis.blocks_pg (ID, geom, block)
VALUES (1, GeomFromText('POLYGON((52 28, 58 28, 58 23, 52 23,
52 28)', 2227), 'block');
```
Working with spatial data in PostgreSQL

• Register with Geodatabase

```bash
// Geodatabase
sdelayer -o register -l blocks_st,shape -e a
 -C objectid,SDE -t ST_GEOMETRY -i esri_sde
-D mydb -u gis -p gis
```

```bash
// Geodatabase
sdelayer -o register -l blocks_pg,geom -e a
 -C objectid,SDE -t PG_GEOMETRY -i esri_sde
-D mydb -u gis -p gis
```
Working with spatial data in PostgreSQL

- Querying spatial attributes

```sql
SELECT st.objectid, st_astext(shape)
FROM map.blocks_st st
WHERE st_astext(sde.st_intersection(sde.st_geometry('polygon ((52 28,58 28,58 23, 52 23,52 8)) ,1),st.shape)::text
NOT LIKE '%EMPTY%';
```

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>SHAPE_WKT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>POLYGON ((2217028.84 399516.70, 2217028.84 399507.82, 2217039.12 399507.82, 2217039.12 399516.70, 2217028.84 399516.70))</td>
</tr>
</tbody>
</table>
Working with SQL Server 2008 Spatial Types
Spatial data in SQL Server 2008

• Microsoft introduces two new Spatial data types

  \textit{GEOMETRY}
  
  – XY coordinates on a plane
  – Supports geographic and projected coordinate systems
  – OGC compliant

  \textit{GEOGRAPHY}
  
  – Latitude, longitude coordinates on an ellipsoid
  – Supports only geographic coordinate system

  Spatial reference must match SQL Server-supported EPSG codes defined in sys.spatial_reference_systems

• Implemented as .NET CLR (UDT) data type

• Available in all 2008 editions except Compact edition
### Comparing Geometry and Geography types

<table>
<thead>
<tr>
<th>ASPECT</th>
<th>GEOMETRY</th>
<th>GEOGRAPHY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line interpolation</td>
<td>Cartesian</td>
<td>Great Circle</td>
</tr>
<tr>
<td>Spatial Domain</td>
<td>Defined by coordinate system</td>
<td>Sphere (Earth)</td>
</tr>
<tr>
<td>Units from calculations</td>
<td>Defined by coordinate system – same as data units</td>
<td>Defined by coordinate system – meters or feet</td>
</tr>
<tr>
<td>Supported coordinate systems</td>
<td>Any SRID SRID has no inherent meaning, but is required for comparisons</td>
<td>Defined in sys.spatial_reference_systems table</td>
</tr>
</tbody>
</table>
• 4-level tesselated grid hierarchy
  • Adjustable

• Created by ArcGIS using feature class extent and dbtune parameters B_MS_SPINDEX and A_MS_SPINDEX
Spatial index usage

- SQL Server Query Optimizer determines whether spatial index is used
  - Improvements to optimizer with regard to spatial indexes announced for SQL Server 2008 SP1
  - Behavior of optimizer is still evolving
  - Currently difficult to make recommendations on spatial index settings for different mixes of features

- Sp_help.spatial_geometry/geography_index
  - Describes efficiency of spatial index for a given filter shape
Creating a SQL Server spatial index

**GEOGRAPHY TYPE**

```sql
CREATE SPATIAL INDEX spidx_geography
    ON world_rivers (geography_col)
    USING GEOGRAPHY_GRID WITH (GRIDS = (LOW, LOW, MEDIUM, HIGH),
    CELLS_PER_OBJECT = 64);
```

**GEOMETRY TYPE**

```sql
CREATE SPATIAL INDEX spidx_geometry
    ON parcels (geometry_col)
    USING GEOMETRY_GRID WITH (BOUNDING_BOX = (0, 0, 500, 200),
    GRIDS = (MEDIUM, MEDIUM, MEDIUM, HIGH),
    CELLS_PER_OBJECT = 64);
```
-- Select all rivers that intersect Sonora

```sql
SELECT r.name, r.shape.STLength()/1000 length_km FROM gdb.rivers r WHERE (SELECT s.shape FROM gdb.states s WHERE name = 'Sonora').STIntersects(r.Shape) = 1
```

<table>
<thead>
<tr>
<th>name</th>
<th>length_km</th>
<th>shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bavispe</td>
<td>453.956548012084</td>
<td>0x61000000010427...</td>
</tr>
<tr>
<td>Colorado</td>
<td>99.39311308932587</td>
<td>0x61000000010409...</td>
</tr>
<tr>
<td>Moctezuma Yaqi</td>
<td>201.929099888165</td>
<td>0x6100000001040D...</td>
</tr>
<tr>
<td>Papigochic</td>
<td>452.549465466336</td>
<td>0x61000000010421...</td>
</tr>
<tr>
<td>Rio Fuerte</td>
<td>438.032117335314</td>
<td>0x61000000010422...</td>
</tr>
<tr>
<td>Rio Sonora</td>
<td>365.48638979468</td>
<td>0x61000000010413...</td>
</tr>
<tr>
<td>Yaqui</td>
<td>263.633008413388</td>
<td>0x61000000010416...</td>
</tr>
</tbody>
</table>
Registering existing SQL Server tables with spatial columns

• Registration: storing metadata about a table so that ArcGIS knows the characteristics of the spatial attribute
  • Tables created with ArcGIS are already registered
  • Tables created with SQL Server or other tools must be manually registered

```bash
sdelayer -o register -l blocks,geog_col -e an+c -t GEOGRAPHY -C objectid,sde -G 4326 -k GEOGRAPHY -P HIGH ...
```
Registration considerations

• Registration does not add a spatial index or calculate a layer extent
  – Manually create your own index, or move layer in and out of load_only_io mode
  – Use `sdelayer -o alter` to calculate an extent

• Registration does not check for valid geometries
  – Use `STIsValid` and `MakeValid` methods

• Registration pre-requisites
  – Single spatial column
  – Single entity type (point, line or polygon)
  – Single coordinate reference (SRID)
Registration and SRID

• What is an SRID
  – ArcGIS: complete spatial reference (sde_spatial_references)
    • offsets, scale, units, cluster tolerance, coordinate system
  – Geography: coordinate system (sys.spatial_reference_systems)
  – Geometry: Arbitrary but required

• Registering a table with existing features
  – Existing SQL Server SRID is recorded in sde_spatial_references (auth_srid)
Registering empty tables

• ArcGIS sets SQL Server SRID
  – **Geography**: epsg code from `sys.spatial_reference_systems`
  – **Geometry**: epsg code from `sys.spatial_reference_systems` (if any)
    OR ArcGIS SRID

• **Workflow: Geography**
  – Provide –G with valid coordinate reference
  – If no –G specified default is 4326 WGS84

• **Workflow: Geometry**
  – Using SQL, insert single feature with desired SRID
  – Provide –G with valid coordinate reference
  – Delete dummy feature after registration

• Alternatively, use –R option to use existing ArcGIS spatial reference
Working with Oracle stGeometry
Knowledge Base

Many articles available
Migration options for moving to a spatial type

- ArcGIS 9.3 default storage with Oracle is st_geometry
- Geoprocessing tool for converting non-versioned and versioned feature classes in-line
Oracle Export/Import with st_geometry and spatial indexes

• *ArcGIS 9.3*
  – Spatial index no longer exported, but created during the import

See Knowledge Base article: 34342
Oracle partitioning and st_geometry

- Supports range partitioning with global or local spatial indexes

- ArcGIS 9.3 provides further support for creating partitioned spatial indexes via the dbtune
  - Future development work for providing partitioned statistics

See Knowledge Base article: 34119
Index scans or Full table scans — Clustering

- Represents how well the table is clustered based on the indexed attribute

- For the rows where owner = ‘BROWN’
  - Are the rows stored in the same data block
  - Or are the rows dispersed among many data blocks

- When clustered, less i/o is required to retrieve the row
### Example: Clustered table

#### Table Extent

<table>
<thead>
<tr>
<th>Rows</th>
<th>149</th>
<th>151</th>
<th>152</th>
</tr>
</thead>
</table>

#### Key Column Values

- 149
- 151
- 152

#### Index

- Leaf Block

<table>
<thead>
<tr>
<th>Key Column Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
</tr>
<tr>
<td>151</td>
</tr>
<tr>
<td>152</td>
</tr>
</tbody>
</table>
Example: Non-clustered table

Key Column Values

149
151
152

Index Leaf Block

Table Extent

Row 149
Row 152
Row 151
Clustering applies to spatial attributes

• Very important when the majority of queries are spatial
  – Every map display…

• SDE administration command for non-versioned layers

  sdeexport -o create -t <table_name> -f <export_file>
  [-i <service>] [-s <server_name>] [-D <database>]
  -u <DB_User_name> [-p <DB_User_password>] [-N] [-q]

• Able to use SQL to spatially cluster data

  See Knowledge Base article: 32423
  See Knowledge Base article: 33341
st_geometry synonyms

- Best practice when developing SQL applications is to fully qualify all objects

```
SELECT owner, sde.st_astext(shape) AS GEOMETRY
FROM tb.parcles
WHERE sde.st_envintersects(shape,10,10,20,20) = 1
```

See Knowledge Base article: 34004
See Knowledge Base article: 34328
Oracle 11g and 10.2.0.4 prohibit the creation of public synonyms named st_* \((\text{st\_geometry}, \text{st\_contains}, \text{st\_relate}...)\)

- ArcGIS 9.3 no longer creates public synonyms for types and operators for new installations
  - Application developers must fully qualify all references to types and operators

- Upgrading from an ArcGIS 9.2 instance will not drop existing public synonyms
  - Application developers can continue to leverage existing public synonyms
Power of aggregation

• New functionality with ArcGIS 9.3 for Oracle
  – st_union_all
  – st_intersection_all
  – st_convexhull_all

```sql
SELECT owner, sde.st_astext(
    sde.st_union_all(shape)) AS GEOMETRY
FROM tb.centerlines
WHERE street_name = 'CENTER ST';

INSERT INTO lightning_areas SELECT oid.nextval,
    sde.st_convexhull_all(shape)
FROM lightning_strikes
WHERE strike_date = TO_DATE('03-15-08', 'MM-DD-YYYY');
```
Area and length are properties of the st_geometry type can be queried directly

- Verses deriving the property from the geometry directly via an operator

```sql
SELECT name, species, a.shape.area AS AREA
FROM forrest_stands a
WHERE a.shape.area > 100000

// Create the index...
CREATE INDEX forrest_area
ON forrest_stands (shape.area);
```
Working with relational operators
Relational operators

Difference between a.shape and b.shape

Is it… \texttt{st\_contains(a.shape, b.shape)}

or

\texttt{st\_contains (b.shape, a.shape)}

• Review the operator definition

• Check join aliases (a table, b table)

• Impacts the optimizer and access path

• Test and \textit{know the results}!
st_distance

• Is **NOT** a relational operator which leverages the spatial index

• It is a function for computing distance between two objects

**Question:**
“Locate all wells within 5 miles from a given location and order the results based on the distance…”
st_distance

- Inefficient approach…

```sql
SELECT name, volume, st_distance(shape, st_point(125,350,3)) AS DISTANCE
FROM wells
WHERE st_distance(shape, st_point(125,350,3)) < 5
ORDER BY distance
```

- Results in a full table scan of wells, calculating the distance between each well and the input geometry to detect if the distance is less than 5 miles
st_distance

- Efficient approach...

```
SELECT name, volume, st_distance(shape,  
    st_point(125,350,3)) AS DISTANCE
FROM wells
WHERE st_intersects(shape,  
    st_buffer(st_point(125,350,3),5)) = 1
ORDER BY distance
```
How st_intersects and st_buffer interacts

- A very common access path for finding relationships between objects is to buffer an input shape and use the output geometry as input to st_intersects

Question: “I want to discover all single family residential parcels with an appraisal value less than 125K and are within a ½ mile distance from a bus stop…”
How st_intersects and st_buffer interacts

- Efficient access path (but still expensive…)

```
SELECT a.address, a.pin, st_distance(a.shape,b.shape)
FROM parcels a, busstops b
WHERE a.appraisal < 125000 AND a.zoning = 'SFR'
AND st_intersects(b.shape,
    st_buffer(a.shape,2640)) = 1
```

- Initial access path is via a composite attribute index on appraisal and zoning returning each parcel’s shape as input to `st_buffer`, which is than input to `st_intersects` and busstops’ spatial index
- For each candidate two spatial operations
How are geometries passed to operators...

- **Step 1:** Access path via the parcel’s composite index for the predicate filter to obtain the parcel geometry

WHERE a.appraisal < 125000 AND a.zoning = ‘SFR’

<table>
<thead>
<tr>
<th>Index</th>
<th>Parcel’s table</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWID</td>
<td>OBJECTID</td>
</tr>
<tr>
<td>AADA...</td>
<td>136072</td>
</tr>
<tr>
<td>AABA...</td>
<td>852018</td>
</tr>
<tr>
<td>AABC...</td>
<td>29375</td>
</tr>
<tr>
<td>AABC...</td>
<td>482190</td>
</tr>
<tr>
<td>AADA...</td>
<td>136072</td>
</tr>
<tr>
<td>AABA...</td>
<td>852018</td>
</tr>
<tr>
<td>AABC...</td>
<td>29375</td>
</tr>
<tr>
<td>AABC...</td>
<td>482190</td>
</tr>
<tr>
<td>AADA...</td>
<td>136072</td>
</tr>
<tr>
<td>AABA...</td>
<td>852018</td>
</tr>
<tr>
<td>AABC...</td>
<td>29375</td>
</tr>
<tr>
<td>AABC...</td>
<td>482190</td>
</tr>
<tr>
<td>AADA...</td>
<td>136072</td>
</tr>
<tr>
<td>AABA...</td>
<td>852018</td>
</tr>
<tr>
<td>AABC...</td>
<td>29375</td>
</tr>
<tr>
<td>AABC...</td>
<td>482190</td>
</tr>
</tbody>
</table>
How are geometries passed to operators...

• Step 2: Each geometry is passed individually to the `st_buffer` operator

```
st_buffer(a.shape, 2640)
```

<table>
<thead>
<tr>
<th>ROWID</th>
<th>OBJECTID</th>
<th>SHAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AADA…</td>
<td>136072</td>
<td></td>
</tr>
<tr>
<td>AABA…</td>
<td>852018</td>
<td></td>
</tr>
<tr>
<td>AACD…</td>
<td>854211</td>
<td></td>
</tr>
<tr>
<td>AABC…</td>
<td>29375</td>
<td></td>
</tr>
<tr>
<td>AABC…</td>
<td>482190</td>
<td></td>
</tr>
</tbody>
</table>
How are geometries passed to operators...

- Step 3: The buffered shape becomes the input to `st_intersects`

```plaintext
AND st_intersects(b.shape,
st_buffer(a.shape, 2640)) = 1
```

- Using the buffered shape’s envelope as the primary filter to search busstop’s spatial index
How are geometries passed to operators…

- Step 4: The envelope is applied to the spatial index to detect all intersecting bus stops.
How are geometries passed to operators…

• Step 5: Each envelope intersecting bus stops is passed to the \texttt{st\_intersects} function to return their relation
  – (true or false)

Returns true, has relation

Returns false, no relation
How to further improve performance with relational operators

- Secondary filtering...

**Question:**

“Which parcels intersect the selected neighborhood?”

```sql
SELECT COUNT(*)
FROM parcels a, neighborhoods b
WHERE b.name = 'WESTWOOD' AND
st_intersects(a.shape, b.shape) = 1
```

401,522 Parcels
52 Neighborhoods
Primary Filter

- Neighborhoods envelope is used as the primary search filter to discover all candidate parcels, via its spatial index

17,584 Candidate Parcels
Secondary Filtering

- Using parcels spatial index grid properties, categorize all exterior, interior and boundary grid cells.
Secondary Filtering
Exterior conditions

- All parcels in exterior grid cells are immediately removed from the candidate list
  - Relation equals false (0)

5,989 Exterior Parcels
Secondary Filtering

Interior conditions

- All parcels in interior grid cells are immediately added to the result set
  - Relation equals true (1)

8,246 Interior Parcels
Secondary Filtering
Boundary conditions

- All parcels intersecting boundary cells require additional filtering to further refine the candidate list

4,965 Boundary Parcels
Secondary Filtering
Boundary Conditions

• Boundary cells are further tessellated and each candidate parcel is evaluated for exterior, interior and boundary intersections.
• Secondary filtering reduces the number of relational comparisons between the input geometry and each candidate geometry

  – And *significantly* decreases user response time...

  – **Primary filtering:** 16.38 seconds
    • 17,584 relational comparisons

  – **Secondary filtering:** .58 seconds
    • 1,237 relational comparisons

10,403 Parcels in the final result set
Comparing Spatial Data Types

(it's like comparing apples to oranges)
How does one compare Geometry Storage types based on “out-of-the-box” tuning and default storage configurations?
Comparing Spatial Data Types

ArcGIS

DB2

SQL Server

Oracle

www.ideageospatial.com
Comparing Spatial Data Types

SAME Cold/Hot Buffer Cache Rate…?
Comparing Spatial Data Types

Reason...BLOB is not CACHE’d

Oracle’s Default BLOB configuration is NOCACHE

DBTUNE - STORE AS (CHUNK 8K CACHE ENABLE
STORAGE IN ROW PCTVERSION 1)

See Knowledge Base article: 33428
Comparing Spatial Data Types

”...It is interesting to see how poorly Oracle ST Geometry performed throughout the tests. It is possible that ArcSDE has not yet been optimized for this new storage format. That fact that it does not use LOB storage should offer a benefit, though it is a custom data type and Oracle may not yet be optimized to deal with custom data types, perhaps the deployment itself may need further optimization...” IdeaGeoSpatial

Performance Difference IS...ROW Storage

DBTUNE - STORE AS (CHUNK 8K CACHE ENABLE STORAGE IN ROW PCTVERSION 1)
Comparing Spatial Data Types

ArcGIS using App Server or Direct Connect use Compressed Binary format.
• Questions…

• We will be available in the Tech Talk area
  – Outside the presentation room