Best Practices for using SQL

Thomas Brown

Kevin Watt
Please!
turn OFF cell phones
and refrain from using
flash photography
Our assumptions

• Prerequisites: Experience writing SQL

• Our expectations...
  – You have an understanding of the geodatabase
  – You understand spatial relationships
  – You have experience with SQL concepts

• Our objective...
  – Introduce the value of using spatial types
  – Provide insight for understanding SQL performance
  – Highlight new functionality with st_geometry for Oracle
Today’s agenda

- Efficient SQL with ArcObjects
- Working with multi-version views
  - Creating, editing and obtaining row_id values
- Working with spatial types
  - What is a spatial type
  - Accessing spatial data
  - Working with relational operators
- Think like an Optimizer
  - Selectivity and cardinality
  - When is the spatial index used
  - Spatial clustering
- Writing a complex spatial query
Efficient SQL with ArcObjects
Efficient SQL with ArcObjects

• As ArcObject developers you are responsible for writing efficient SQL
  
  – Fetch only the necessary subfields
  – Join the appropriate tables
  – Define efficient where clauses
  – Set correct postfix clauses
Efficient SQL with ArcObjects

• Example: table join with a complicated where clause

```csharp
IQueryDef queryDef = featureWorkspace.CreateQueryDef();
queryDef.Tables = "ElectricStation, CircuitSource, CableJunction";
queryDef.SubFields = "ElectricStation.StationName,
                     ElectricStation.StationAbbreviation, CableJunction.ObjectID";
queryDef.WhereClause = "((ElectricStation.CircuitID LIKE 'A-235%' +
                        "AND SUBSTR(ElectricStation.CircuitID,5+1,1) not between '0' and '9')" +
                        "OR (ElectricStation.CircuitID = 'A-235'))" +
                        "AND ElectricStation.StationTypeCode = 'GEN'" +
                        "AND CircuitSource.ObjectID = ElectricStation.CircuitSourceObjectID" +
                        "AND ElectricStation.ObjectID = CircuitSource.ElectricStationObjectID" +
                        "AND CableJunction.DeviceObjectID = ElectricStation.ObjectID" +
                        "AND CableJunction.DeviceType = 'ENG'" +
                        "OR INSTR(UPPER(CircuitSource.PhaseDesignation),'123') > 0)";

ICursor cursor = queryDef.Evaluate();
IRow row = cursor.NextRow();
```
Efficient SQL with ArcObjects

- Watch out for DBMS operators which might require full table scans
  - SUBSTR()
  - INSTR()
  - UPPER()

- Utilize function based indexes when applicable

- Sometimes efficient SQL means executing multiple single table statements and generate the final result set within the client application
Efficient SQL with ArcObjects

- **IWorkspace.ExecuteSQL**
  - Execute arbitrary SQL statements
  - DDL or DML statements (but can not return a result set)
  - Stored procedures

```csharp
IWorkSpace.ExecuteSQL 'BEGIN <stored_procedure_name> ("<arguments>"); END;

//Within your procedure
err_num := SQLCODE;
erm_msg := SUBSTR(sqlerrm, 1, 100);
INSERT INTO <temporary_table> VALUES (err_num, err_msg);
COMMIT;

//Use ArcObjects to check the return code
IQueryDef queryDef = featureWorkspace.CreateQueryDef();
queryDef.Tables = "<temporary_table>"
queryDef.SubFields = "err_num, err_msg"
IRow row = cursor.NextRow();
```
Working with multi-versioned views
Working with multi-versioned views

• For enterprise 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 multi-versioned 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 multi-versioned views

- DBMS procedures for editing a versioned geodatabase and multi-versioned views

---

//Oracle

```sql
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

```sql
exec sde.edit_version ('tomb.PROPOSED_SUBDIVISION', 1)
exec dbo.edit_version ('tomb.PROPOSED_SUBDIVISION', 2)
```
Working with multi-versioned views

- DBMS procedures for obtaining row_id values

//Oracle

```sql
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

```sql
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
```
Working with multi-versioned views

• Do NOT...
  – Update the objectid (row_id) value
  – 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
  – Update attributes which define geodatabase behavior
    • Enabled/Disabled attributes
    • Ancillary attributes
    • Weight attributes
    • Subtypes
Working with spatial types
Working with Spatial Types

It’s about…

Spatial Types
  - What is a “Spatial Type”
  - Accessing Spatial Data
  - Working with Relational Operators

It’s not about…

Accessing geodatabase objects using SQL
Comparing Spatial Types
Geodatabase / ArcSDE geometry storage formats

Relational or Object Relational

Storage formats vary by DBMS

– Oracle
  • SDE Binary (LONG RAW) and (LOB)  Relational / Relational
  • ST_Geometry  Object Relational
  • SDO_Geometry  Object Relational

– SQL Server
  • SDE Binary  Relational

– DB2
  • Spatial Extender  Object Relational

– Informix
  • Spatial Data Blade  Object Relational
Geometry storage formats

Relational

SELECT id, a.shape, f.fid
FROM Buildings a, F1 f, S1 s
WHERE s.gx <= :1 AND s.gx >= :2 AND s.gy <= :3 AND s.gy >= :4
    AND minx <= :5 AND miny <= :6 AND maxx >= :7 AND
    maxy >= :8 AND f.fid = s.sid AND a.shape = f.fid
Geometry storage formats

Object Relational

<table>
<thead>
<tr>
<th>Buildings</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
</tbody>
</table>

```
SELECT b.id, b.shape
FROM Buildings b
WHERE ST_EnvIntersects (b.shape,10, 10, 50, 50) = 1;
```

Easier to work with in SQL than Relational
What is a Spatial Type...

- Object types for geometry data
  - ST_Geometry, SDO_Geometry

- Spatial Index

- Relational and Geometry Operators
  - ST_Relate
  - ST_Intersects
  - ST_Buffer...
Geometry object type properties

- Numpts – # of points
- Entity – Geometry type
- MinX – Minimum X
- MinY – Minimum Y
- MaxX – Maximum X
- MaxY – Maximum Y
- MinZ – Minimum Z
- MaxZ – Maximum Z
- MinM – Minimum Measure
- MaxM – Maximum Measure
- Area – Area of polygon
- Len – Length of line/polygon
- SRID – Spatial Reference
- Geometry – coordinates
ST_GEOMETRY object model

ST_GEOMETRY

ST_Point

ST_Curve

ST_Surface

ST_GeomCollection

ST_LineString

ST_Polygon

ST_MultiPoint

ST_MultiCurve

ST_MultiSurface

ST_MultiLineString

ST_MultiPolygon

OpenGIS Simple Features Specification for SQL

ST_GEOMETRY constructor functions

• Well-Known Binary
  – ST_AsBinary

• Well-Known Text
  – ST_AsText

• ESRI ShapeFile
  – ST_AsShape
ST_GEOMETRY constructors

- Well-Known Binary - ST_AsBinary

BEGIN
    FOR item IN
        (SELECT area, ST_AsBinary(shape)
         FROM Buildings
         WHERE ST_Area(shape) > 80000
         )
    LOOP
        dbms_output.put_line('Area: '||item.area||' ST_AsBinary Len: '||dbms_lob.GETLENGTH(item.shape1));
    END LOOP;
END;

Area: 92116.13714 ST_AsBinary Len: 237
# ST_GEOMETRY constructors

- **Well-Known Text** - ST_AsText

```sql
SELECT area, ST_AsText(shape) as shape_wkt
FROM buildings
WHERE ST_Area(shape) < 100;
```

<table>
<thead>
<tr>
<th>AREA</th>
<th>SHAPE_WKT</th>
</tr>
</thead>
<tbody>
<tr>
<td>91.362</td>
<td>POLYGON ((2217028.84 399516.70, 2217028.84 399507.82, 2217039.12 399507.82,</td>
</tr>
<tr>
<td></td>
<td>2217039.12 399516.70, 2217039.12 399516.70, 2217028.84 399516.70))</td>
</tr>
</tbody>
</table>
ST_GEOMETRY constructors

- ESRI ShapeFile - ST_AsShape

BEGIN
  FOR item IN
    (SELECT area, ST_AsShape(shape)
     FROM Buildings
     WHERE ST_Area(shape) > 80000
    )
  LOOP
    dbms_output.put_line('Area: '||item.area||' ' || 'ST_AsShape Len:'||dbms_lob.GETLENGTH(item.shape1));
  END LOOP;
END;

Area: 92116.13714     ST_AsShape Len: 237
Type checking through inheritance

- Strong type checking is used to enforce type correctness at the subtype level

```sql
SQL> CREATE TABLE test_load (id INTEGER, shape ST_Polygon);

SQL> INSERT INTO test_load
    values (10, ST_Polygon ('polygon ((10 10,10 20,20 20,
                                20 10,10 10))' ,1));

SQL> INSERT INTO test_load (id, shape)
    values (2, ST_Point (50, 50, 1));
```

ST_GEOMETRY object model

- ST_Point
- ST_Curve
- ST_Surface
- ST_GeomCollection

- ST_LineString
- ST_Polygon

- ST_MultiPoint
- ST_MultiCurve
- ST_MultiSurface

- ST_MultiLineString
- ST_MultiPolygon
Type checking through inheritance

• Strong type checking is used to enforce type correctness at the subtype level

SQL> CREATE TABLE test_load (id INTEGER, shape ST_Polygon);

SQL> INSERT INTO test_load
   values (10, ST_Polygon ('polygon ((10 10,10 20,20 20,
   20 10,10 10))' ,1));

SQL> INSERT INTO test_load (id, shape)
   values (2, ST_Point (50, 50, 1));

ERROR at line 1:
ORA-00932: inconsistent datatypes: expected SDE.ST_POLYGON
   got SDE.ST_POINT
What is a Spatial Type...

- Object types for geometry data
  - ST_Geometry, SDO_Geometry

- Spatial Index
  - Vendor-specific Format

- Relational and Geometry Operators
  - ST_Relate
  - ST_Intersects
  - ST_Buffer...
Spatial Index

- Uses DBMS Extensible Indexing

- Search Algorithm - Grid or RTree Indexing

- Associated to ST_Geometry type and Operators
  - Provides access path information to the optimizer
What is a Spatial Type...

- Object types for geometry data
  - ST_Geometry, SDO_Geometry

- Spatial Index

- Relational and Geometry Operators
  - ST_Relate
  - ST_Intersects
  - ST_Buffer...
Spatial Type Operators

Geometry

“returns a GEOMETRY or property from the difference, comparison or inspection of geometries”

Relational

“…returns TRUE
if SHAPE1 has a spatial relationship {intersect, touches, contains, within,…} with SHAPE2
else
returns FALSE”
Geometry Operators

- **ST_Buffer**
  - Return Geometry whose distance from Geom1 is less than or equal to a specified distance

- **ST_IsClosed**
  - Return TRUE if Curve or MultiCurve is closed

- **ST_AsText**
  - Return Well-Known Text from Geom

- **ST_Intersection**
  - Return Geometry that represents the point set intersection of Geom1 and Geom2

- **ST_Envelope**
  - Return Geometry that represents the rectangle of Geom1

- **ST_Union**
  - Returns Geometry that represents the point set union of Geom1 and Geom2
Geometry Operators

```
SELECT b.building_name
FROM buildings b, parcels p
WHERE p.pid = 45078
  AND ST_Intersects(ST_Buffer(p.shape, 1000), b.shape) = 1
  AND ST_Area(b.shape) > 4000;
```

“select building names from Buildings that intersect Parcels having a 1000 meter buffer around parcel 45078 and are greater than 4000 square feet”

More on Geometry Operators visit

<table>
<thead>
<tr>
<th>Relational Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_Contains</td>
<td>TRUE if Geom1 contains Geom2</td>
</tr>
<tr>
<td>ST_Crosses</td>
<td>TRUE if Geom1 crosses Geom2</td>
</tr>
<tr>
<td>ST_Disjoint</td>
<td>TRUE if Geom1- Geom2 do not intersect</td>
</tr>
<tr>
<td>ST_Equals</td>
<td>TRUE if Geom1- Geom2 are the same</td>
</tr>
<tr>
<td>ST_Intersects</td>
<td>TRUE if Geom1- Geom2 intersect</td>
</tr>
<tr>
<td>ST_OrderingEquals</td>
<td>TRUE if Geom1- Geom2 are the same and in the same order</td>
</tr>
<tr>
<td>ST_Overlap</td>
<td>TRUE if Geom1- Geom2 overlap and resultant geometry is the same type</td>
</tr>
<tr>
<td>ST_Relate</td>
<td>TRUE if Geom1- Geom2 meet the conditions of the DE-9IM matrix**</td>
</tr>
<tr>
<td>ST_Touches</td>
<td>TRUE if Geom1- Geom2 touch</td>
</tr>
<tr>
<td>ST_Within</td>
<td>TRUE if Geom1 is completely within Geom2</td>
</tr>
</tbody>
</table>
Relational Operators

```sql
SELECT b.building_name
FROM buildings b, parcels p
WHERE p.zipcode = 90210
  AND ST_Contains(p.shape,b.shape) = 1
  AND ST_Area(b.shape) > 4000;
```

*ST_Contains* returns TRUE if *p.shape* contains *b.shape*

“*select all buildings names from Buildings that are in Parcel ZipCode 90210 and completely contained in the parcel boundaries from 90210 and are greater than 4000 square feet* ”
Relational Operators

Remember…

- Difference between a.shape and b.shape
- Why FALSE “0” isn’t the opposite of TRUE “1”
- Use ST_Relate to solve simple and complex spatial relationships
- Avoid overloading predicate with multiple spatial operators
Relational Operators

Difference between a.shape and b.shape

Is it... $\text{ST\_Contains} (a\text{.shape}, b\text{.shape})$

or

$\text{ST\_Contains} (b\text{.shape}, a\text{.shape})$

• Review the operator description

• Check join aliases

• Test against local test case and *know the results!*
Why FALSE isn’t the opposite of TRUE

• Two layers
  – One layer with two points
  – One layer with two polygons
Group Exercise

• Question 1

- What is the value for COUNT(*)?

```
SELECT COUNT(*) FROM pnts pt, polys pl WHERE st_intersects(pt.shape, pl.shape) = 1;
```
Answer

COUNT = 1
Group Exercise

• Question 2

- What is the value for COUNT(*)?

```
SELECT COUNT(*) FROM pnts pt, polys pl
WHERE st_intersects(pt.shape, pl.shape) = 0;
```
Answer

COUNT = 3
Relational Operators

- **ST_Relate**
  - Compares spatial relationship using Dimensionally Extended 9 Intersection Model (DE-9IM)
  - Performs better than multiple (intersects, touches, ...) predicate filters

Question…

“**I want to find all Census Blocks within a Voting District that have a common linear boundary not inside a Voting District…**”
### Relational Operators - ST_Relate

<table>
<thead>
<tr>
<th>SHAPE B</th>
<th>Interior</th>
<th>Boundary</th>
<th>Exterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interior</td>
<td>F</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Boundary</td>
<td>*</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>Exterior</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SHAPE A</th>
<th>0 Dim</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td>1 Dim</td>
<td>Line</td>
</tr>
<tr>
<td>Exterior</td>
<td>2 dim</td>
<td>Polygon</td>
</tr>
</tbody>
</table>

- T – Intersection(0,1,2)
- F – no Intersection

“Census Blocks and Voting Dist. interiors do not Intersect “F” but have a common Linear “1” boundary “

```sql
SELECT c.shape
FROM census_blocks c, voting_districts v
WHERE v.district = 1394
AND ST_Relate (c.shape, v.shape, 'F***1****') = 1;
```
Relational Operators

For more information on DE-9IM format

“Think” like an optimizer?
What is an optimizer

• The optimizer is the database’s mechanism for using heuristics for choosing an execution plan (access path) for a SQL statement

• During the parsing phase the optimizer determines access paths, join methods and join orders by calculating a COST
How does the optimizer calculate $\textit{COST}$

- During the optimization process, the optimizer discovers the following
  - $\textit{Selectivity}$
  - $\textit{Cardinality}$
**Selectivity**

- Is the fraction of rows from the table which “meet” a predicate’s condition

```
SELECT COUNT(*) FROM parcels WHERE owner = 'BROWN'
```

- Parcels table contains 100 rows
- 10 rows contain a NULL owner
- 85 distinct owner values
**Selectivity**

- **Density equation**
  - \( \frac{1}{\text{number distinct values}} \)
  - \( \frac{1}{85} = .011764 \)

- **Selectivity equation**
  - \( \text{density} \times \left( \frac{\text{number rows} - \text{number nulls}}{\text{number rows}} \right) \)
  - \( \cdot .011764 \times \left( \frac{100 - 10}{100} \right) / 100 = .01 \)
Cardinality

• Computed *cardinality* is the number of rows in the result set after predicates are applied

```
SELECT COUNT(*) FROM parcels WHERE owner = 'BROWN'
```

• *Cardinality* equation
  
  - selectivity * number of rows
  - .01 * 100 = 1

• Crucial for selecting join orders and choice of indexes
Optimizer’s **COST**

- Is calculated by the *optimizer* based upon the amount of *i/o* and *cpu* consumed to perform an operation

- Every operation of a SQL statement has a *cost*
  - Each access step
  - Each predicate filter
  - Each join

- *Cost* of *Full table scan* is the baseline

- *Optimizer* selects the best plan based on lowest *cost*
How does the *optimizer* derive values

- **Table and Index statistics**
  - *It is critical to have accurate statistics*
  - If the statistics do not represent the true data characteristics… the optimizer *will* choose a plan which is inefficient
  - If the data is dynamic, update statistics
Single predicate example

• How would the optimizer determine the best access path for the following statement
  – (attribute owner has a non-unique index)

```sql
SELECT apn, shape FROM parcels WHERE owner = 'BROWN'
```

*Index scan?*

*Full table scan?*
Answer: It depends!

• How many rows are in the table
  – 1 – *Full table scan* (less i/o)
  – 1,000,000 – might still read the entire table

• How many rows in the table where *owner* = ‘BROWN’
  – 1 of 1,000,000 – *Index scan*
  – 900,000 of 1,000,000 – *Full table scan*
  – 50% of the table – *Index scan* or *Full table scan*
  – 25% of the table – *Index scan* or *Full table scan*
Index scan or Full table scan – clustering factor

• 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
Example: Non-clustered table

Key Column Values

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
</tr>
<tr>
<td>151</td>
</tr>
<tr>
<td>152</td>
</tr>
</tbody>
</table>

Index
Leaf Block

Table Extent

Data Block

<table>
<thead>
<tr>
<th>Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>151</td>
</tr>
<tr>
<td>152</td>
</tr>
<tr>
<td>149</td>
</tr>
</tbody>
</table>
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
When should an optimizer use a spatial index

```
SELECT * FROM parcels WHERE st_envintersects(shape,5,5,10,10) = 1
```

• Same question applies…
  – Is it better to perform a *Full table scan* or use the spatial index

• How does the optimizer know which to use
**Full table scan** or use the spatial index?

- Questions to consider
  - How many rows in the table?
  - How large is the search envelope (is it the entire layer)?
  - What is the *selectivity* for the search envelope?
  - What is the *cardinality*?
  - How much *i/o* will be required to use the index?
  - How much *cpu* might be consumed?
  - Is the data spatially clustered?
Gathering spatial statistics

- Execution plans are derived from statistics that are used by types, functions, and domain indexes
  - `dbms_stats.gather_table_stats`
  - Spatial filters are highly selective and spatial indexes are excellent for access paths
  - Spatial domain index statistics stored in `sde.st_geometry_index`
**st Geometry Index statistic attributes**

SQL> describe st_geometry_index

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDXE_NAME</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>UNIQUENESS</td>
<td></td>
<td>VARCHAR2(9)</td>
</tr>
<tr>
<td>DISTINCT_KEYS</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>BLEVEL</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>LEAF_BLOCKS</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>CLUSTERING_FACTOR</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>DENSITY</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>NUM_ROWS</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>NUM_NULLS</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SAMPLE_SIZE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>LAST_ANALYZED</td>
<td></td>
<td>DATE</td>
</tr>
</tbody>
</table>
Leveraging the extensible optimizer

- Utilizing `st_geometry_index` statistics we can assist the optimizer
  - Derive *selectivity*
  - Calculate *cardinality*
  - Set *cost*
When should an optimizer use a spatial index

- Either a **Full table scan** or use the spatial index...

- Which ever cost is less becomes the access path
  - If a **Full table scan** is performed st_envintersects becomes a filter for each row

```sql
SELECT * FROM parcels WHERE st_envintersects(shape,5,5,10,10) = 1
```
How does the optimizer cost operators

- Perform a **Full table scan** and filter each row which intersects the input polygon?

- Use the polygon as an input argument, find all parcels within its envelope (using the spatial index) and then filter each row?

```sql
SELECT * FROM parcels WHERE st_contains(shape, st_geom) = 1
```
How does the optimizer cost operators?

- It is still determined by **cost**

- The **cost** of the *Full table scan* (baseline)
- The **cost** of using the spatial index and the **cost** of the operator

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>190</td>
<td>4 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*1</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>PARCELS</td>
<td>1</td>
<td>190</td>
<td>4 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*2</td>
<td>DOMAIN INDEX (Sel: .000285)</td>
<td>PAR_IDX</td>
<td>1</td>
<td></td>
<td>4 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

2 – access ("SDE"."ST_CONTAINS"("SHAPE,ST_GEOMETRY")=1)
How does the *optimizer* choose which predicates to apply first

```
SELECT * FROM parcels
  WHERE st_envintersects(shape,5,5,10,10) = 1
  AND owner = 'BROWN'
```

- *Optimizer* calculates the following *costs*
  - *Full table scan*
  - Using the spatial index
  - *Index scan* using the owner index
Example: predicate **cost**

- **Full table scan cost** 95
- Using the spatial index **cost** 50
- **Index scan** using the owner index **cost** 12

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>190</td>
<td>12 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>PARCELS</td>
<td>1</td>
<td>190</td>
<td>10 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 2</td>
<td>INDEX SCAN</td>
<td>OWN_IDX</td>
<td>1</td>
<td></td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

- 1 - filter("SDE"."ST_ENVINTERSECTS"("P"."SHAPE",5,5,10,10)=1)
- 2 - access("OWNER"='BROWN')
I know my data better then the *optimizer*...

• There will be cases when the calculated *selectivity* doesn’t accurately represent the data

• ESRI provides the `sde.spx_util` for setting statistics and *selectivity*

  ```
sde.spx_util.set_index_stats
sde.spx_util.set_column_stats
sde.spx_util.set_operator_selectivity
  ```

  – See Knowledge Base article: 32605
Example: Changing *selectivity* for an operator

```
SELECT * FROM buildings
    WHERE st_within(shape, st_geometry) = 1
    AND bld_type = 49
```

- **Current execution plan**

```
<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>190</td>
<td>5 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>BUILDINGS</td>
<td>1</td>
<td>190</td>
<td>5 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 2</td>
<td>DOMAIN INDEX (Sel: .00016)</td>
<td>BUILD_IDX</td>
<td>48</td>
<td></td>
<td>5 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 – filter ("BLD_TYPE=49")
2 - access("SDE"."ST_WITHIN"("SHAPE,ST_GEOMETRY")=1)
Example: Changing *selectivity* for an operator

- **Change st_within *selectivity***
  - From .00016 to .01 (increases the *cardinality*)

```
SQL> exec spx_util.set_operator_selectivity
     ('tb', 'buildings', 'shape', 'st_within', .01);
```

- **New Execution Plan**

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>10</td>
<td>1900</td>
<td>3</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>BUILDINGS</td>
<td>10</td>
<td>1900</td>
<td>3</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 2</td>
<td>INDEX SCAN</td>
<td>BUILD_IDX</td>
<td>10</td>
<td></td>
<td>3</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

```
1 – filter ("SDE"."ST_WITHIN"("SHAPE,ST_GEOMETRY")=1)
2 - access ("BLD_TYPE=49")
```
Statistic type provides

• **Function** *cost*
  – Assists in defining operator precedence

• **Selectivity**

• Domain index *cost*
  – iocost
  – cpucost (dbms_odci.estimate_cpu_units)
  – networkcost (always 0)
  – indexcostinfo *(selectivity)*

• **Without statistics optimizer uses default values**
  – *selectivity* 1% and spatial index *cost* 2
Writing a complex spatial query
Business requirement

• Define your objective prior to writing SQL
  – Identify all parcels within a specified area...
  – A parcel may not contain an existing building...
  – The parcel’s area must be greater than 1 acre...
  – Generate a report listing the parcels by area in descending order...

• Ensures you understand exactly what your query will answer
SQL translation

• Step 1
  – Identify parcels which do NOT have an existing building within a specified area…
  – Don’t fall for the ‘NOT’ trap
  – NOT does not mean using an operator equality 0 (false)

SELECT p.objectid
FROM parcel p, building b
WHERE st_intersects (p.shape, b.shape) = 1
AND st_envintersects (b.shape, 5, 5, 10, 10) = 1

– st_intersects used to identify which parcels intersect buildings
– st_envintersects used for locating all buildings within the specified area
SQL translation

• Step 2
  – Using the result set from step 1 (parcels which intersect buildings) remove parcels from the outer select with NOT IN

```
SELECT par.apn "PARCEL ID", ROUND(par.shape.area) "AREA"
FROM parcel par
WHERE par.objectid NOT IN
  (SELECT p.objectid
   FROM parcel p, building b
   WHERE st_intersects (p.shape, b.shape) = 1
     AND st_envintersects (b.shape, 5, 5, 10, 10) = 1)
AND st_envintersects (par.shape, 5, 5, 10, 10) = 1
```

– **st_envintersects** used for locating all parcels within the specified area
SQL translation

• **Step 3**
  – Add the filter to only return parcels with an area > 1 acre
    
    AND `par.shape.area` > 43560
  
    – *st_geomtery exposes the property of area and length as an attribute*

• **Step 4**
  – Set the ORDER BY clause to report the parcels in descending order by area
    
    ORDER BY `par.area` DESC
SQL translation

• Final statement to execute

SELECT par.apn "PARCEL ID", ROUND(par.shape.area) "AREA"
FROM parcel par
WHERE par.objectid NOT IN
  (SELECT p.objectid
   FROM parcel p, building b
   WHERE st_intersects (p.shape, b.shape) = 1
   AND st_envintersects (b.shape, 5, 5, 10, 10) = 1)
AND st_envintersects (par.shape, 5, 5, 10, 10) = 1
AND par.shape.area > 43560
ORDER BY par.area DESC;
# Optimizer’s execution plan

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>6</td>
<td>2076</td>
<td>20</td>
<td>00:00:01</td>
</tr>
<tr>
<td>1</td>
<td>SORT ORDER BY</td>
<td></td>
<td>6</td>
<td>2076</td>
<td>20</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 2</td>
<td>HASH JOIN ANTI</td>
<td></td>
<td>6</td>
<td>2076</td>
<td>19</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 3</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>PARCEL</td>
<td>7</td>
<td>2387</td>
<td>12</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 4</td>
<td>DOMAIN INDEX (SEL: 214988881)</td>
<td>A24_IX1</td>
<td>1</td>
<td>5</td>
<td>12</td>
<td>00:00:01</td>
</tr>
<tr>
<td>5</td>
<td>VIEW</td>
<td>VW_NSO_1</td>
<td>1</td>
<td>599</td>
<td>6</td>
<td>00:00:01</td>
</tr>
<tr>
<td>6</td>
<td>NESTED LOOPS</td>
<td></td>
<td>1</td>
<td>599</td>
<td>6</td>
<td>00:00:01</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>BUILDING</td>
<td>1</td>
<td>258</td>
<td>3</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 8</td>
<td>DOMAIN INDEX (SEL: 6891841)</td>
<td>A20_IX1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>00:00:01</td>
</tr>
<tr>
<td>9</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>PARCEL</td>
<td>1</td>
<td>341</td>
<td>6</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 10</td>
<td>DOMAIN INDEX (SEL: 6949071)</td>
<td>A24_IX1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

**Predicate Information (identified by operation id):**

- 2 - access("OBJECTID"="$nso_col_1")
- 3 - filter("A"."SYS_NC00026">43560)
- 4 - access("SDEM_TH_ENVIINTERSECTS"("A"."SHAPE",6225268,2295547,6230258,2300178)=1)
- 8 - access("SDE"."ST_ENVINTERSECTS"("B"."SHAPE",6225268,2295547,6230258,2300178)=1)
- 10 - access("SDE"."ST_INTERSECTS"("P"."SHAPE","B"."SHAPE")=1)
Final step: Displaying the result set

<table>
<thead>
<tr>
<th>PARCEL ID</th>
<th>AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>253000001</td>
<td>4702646</td>
</tr>
<tr>
<td>221280002</td>
<td>809172</td>
</tr>
<tr>
<td>221300008</td>
<td>611417</td>
</tr>
<tr>
<td>221232019</td>
<td>415702</td>
</tr>
<tr>
<td>221300006</td>
<td>410333</td>
</tr>
<tr>
<td>221300008</td>
<td>393238</td>
</tr>
<tr>
<td>221260028</td>
<td>382588</td>
</tr>
<tr>
<td>221132015</td>
<td>164814</td>
</tr>
<tr>
<td>221070008</td>
<td>145025</td>
</tr>
</tbody>
</table>

9 rows selected.
What to remember

• **Statistics are critical** for the *optimizer* in determining the optimal execution plan (access path)

• It is important for spatial data to be clustered for improving *i/o*

• For complex queries you have the ability to define operator *selectivity*
Presentation materials

• PowerPoint presentation and code are posted on the conference web site

• EDN – downloads and videos
Other Sessions to Attend

All sessions are in the Catalina/Madera rooms

• **Turbocharged Geodatabase Programming**
  – Wednesday, 21st, 1:00 - 2:15 pm

• **Distributed Geodatabase for Developers**
  – Wednesday, 21st, 4:30 - 5:45 pm

• **Raster Data Management in ArcGIS 9.2**
  – Thursday, 22nd, 8:30 - 9:45 am
Further Questions

• **TECH-TALK**
  – **What:** Opportunity to ask questions and discuss concerns with presenters and other GDB team members
  – **Where:** Room 5
  – **When:** During the next 30 minutes

• **Meet the Teams**
  – **When:** Various times

• **ESRI Developers Network (EDN) website**
Tech Talk Map
Session Evaluations Reminder

Don’t forget the tech talk (Room 5)
Please turn in your session evaluations.

... Thank you