Geography Matters

UC98

San Diego, California

ESRI

EIGHTEENTH ANNUAL INTERNATIONAL USER CONFERENCE

SDE From Design to Deployment

Leo Bynum and Mansour Raad

SDE Overview

Brief Introduction

What is SDE?

Take a standard RDBMS Table and add a *GEOMETRY* column (with X,Y's inside).

This is old news (pre SDE).

Storage is simple.

Attribute query & retrieval easy.

Spatial Queries are More Difficult and Interesting

Spatially and Topologically constrained queries present two problems especially for LARGE data sets.

1) Need very complex and very fast logic to do spatial relationship comparisons.

2) For efficiency, general **area of interest** logic is required to reduce your Search space.

You need:

A GOOD SUPER-FAST TOPOLOGY ENGINE
A GOOD SUPER-FAST SPATIAL INDEXING SCHEME

SDE solves these problems

SDE provides a super fast INTEGER based topology engine

SDE provides a very efficient method of SPATIAL INDEXING (utilizing grids)

Normal SQL Queries

A typical SQL query:

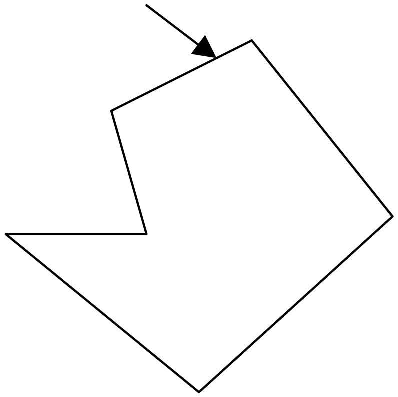
```
SELECT  <COLUMN(S)>
FROM    <TABLE(S)>
WHERE   <COLUMN>  <REALTION> <VALUE>
```

```
select pop96, area from counties
where name = ‘SAN DIEGO’
```

SDE Spatial Queries

SDE will allow you to make these kinds of queries by providing geometric column types and topology relational operators.

SELECT pop96, geometry from counties
where area < 25000 and
geometry [is inside or touching] this



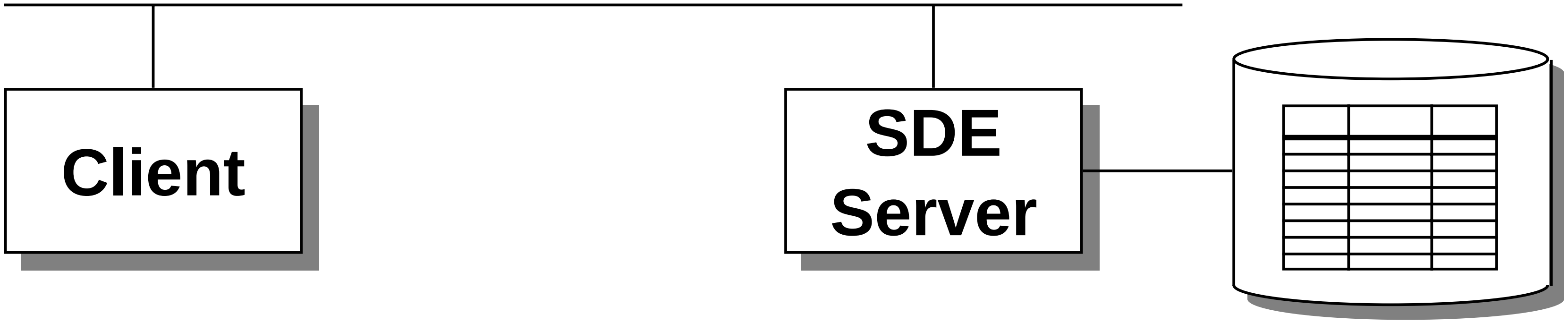
* Note this is conceptual. Actual implementation is programmed in the SDE client

SDE-INFORMIX DataBlade
Spatial Queries

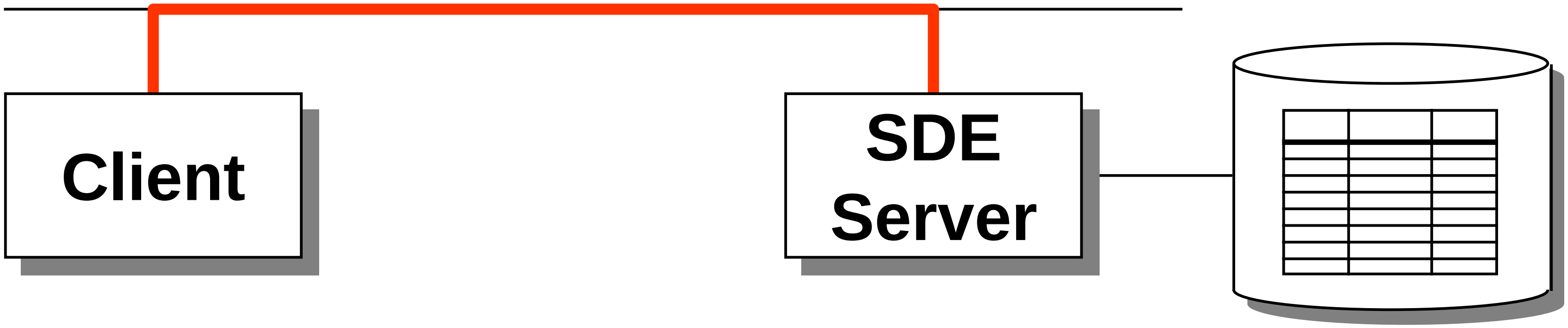
```
Select school_name from schools, toxic_sites
where

WITHIN
    (
      BUFFER( toxic_sites.location, 5.0),
      schools.location
    ) = T
```

* Note: This is the actual syntax for the SDE-INFORMIX DataBlade

Client Server Queries

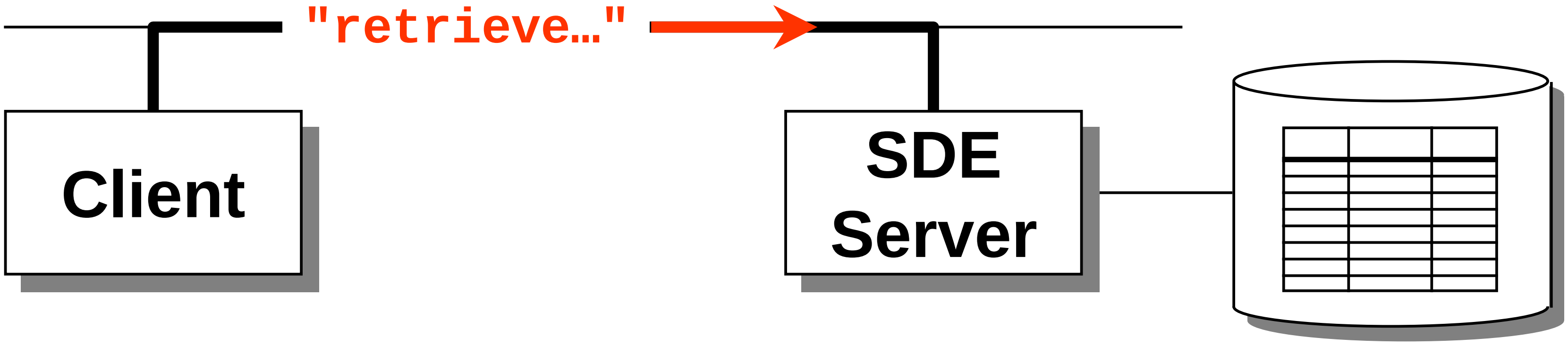


Connection on startup



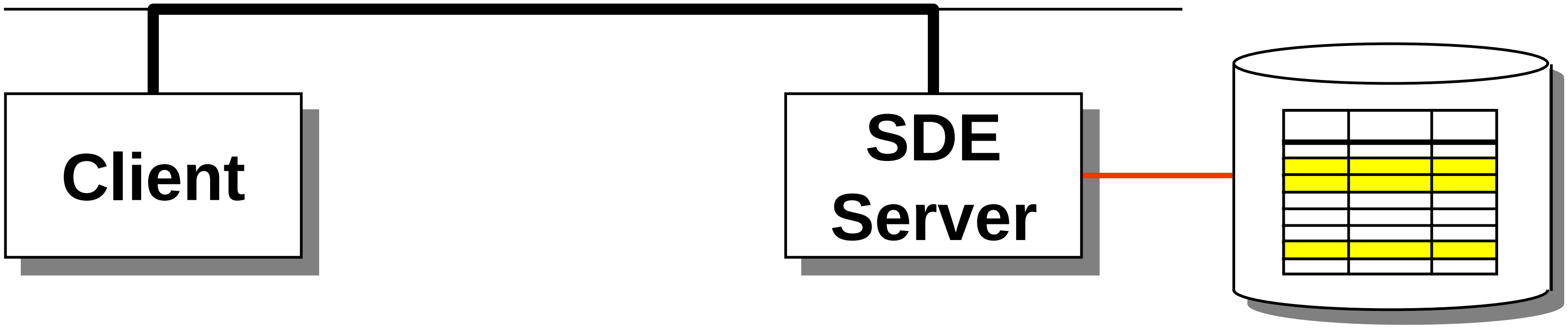
Client connects to server

Query is initiated



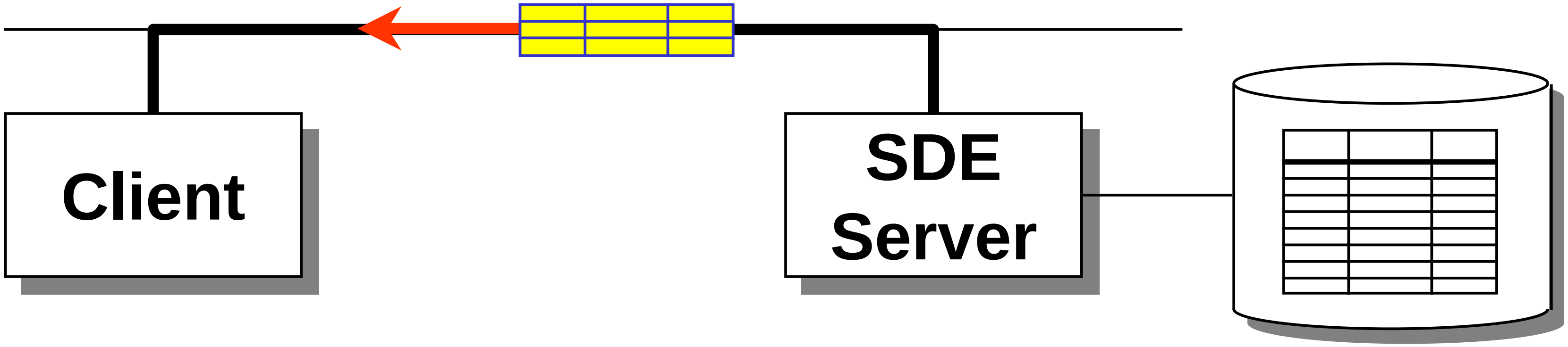
Client submits query to server

SDE/RDBMS do their work



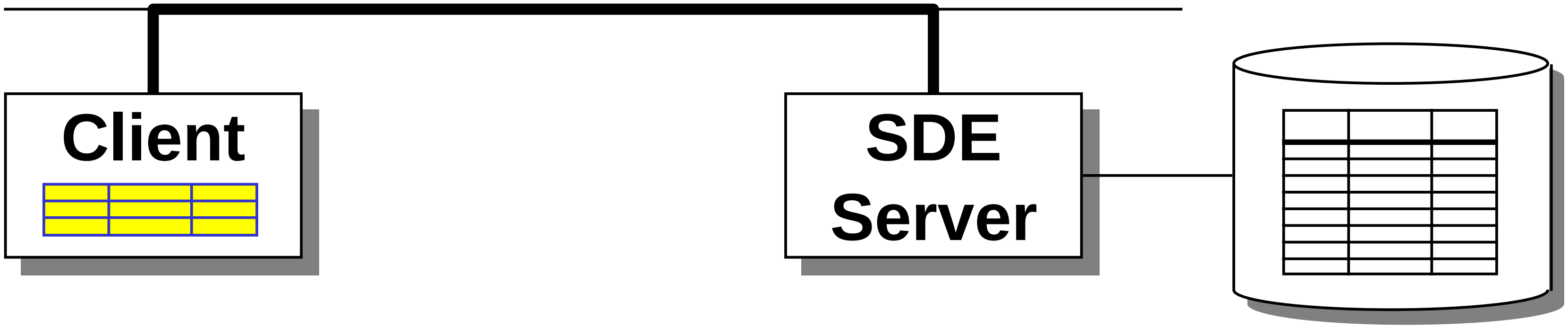
Server finds features satisfying constraints

Results shipped back



Server sends features to client over network

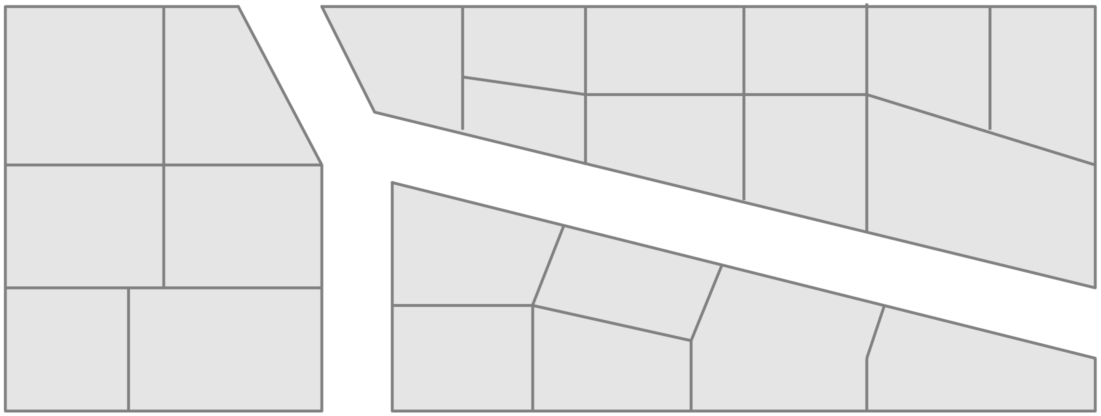
Client Rendering (Drawing)



Client processes one-at-a-time in a loop

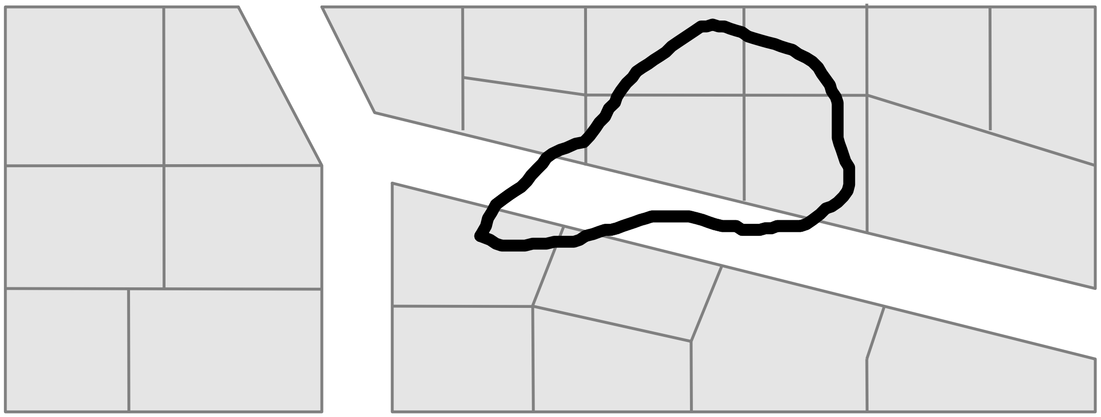
S Table and Grids

Query a Land Parcels Layer



Spatial Query from Client

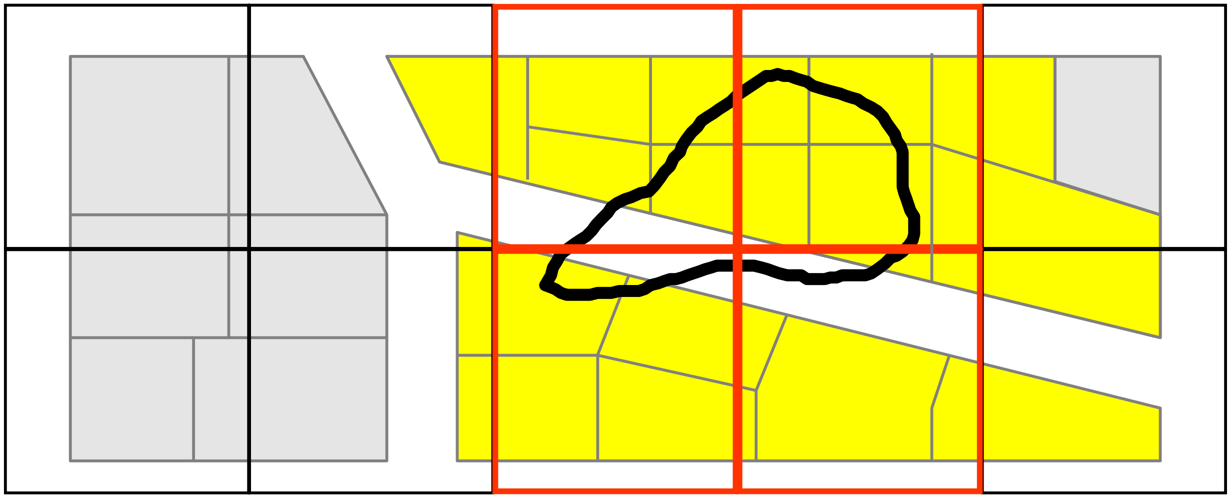
```
"retrieve all land parcels that overlap a
 ruptured storage tank's contamination plume"
```



- Client submits spatially-constrained query

S Table Queried First

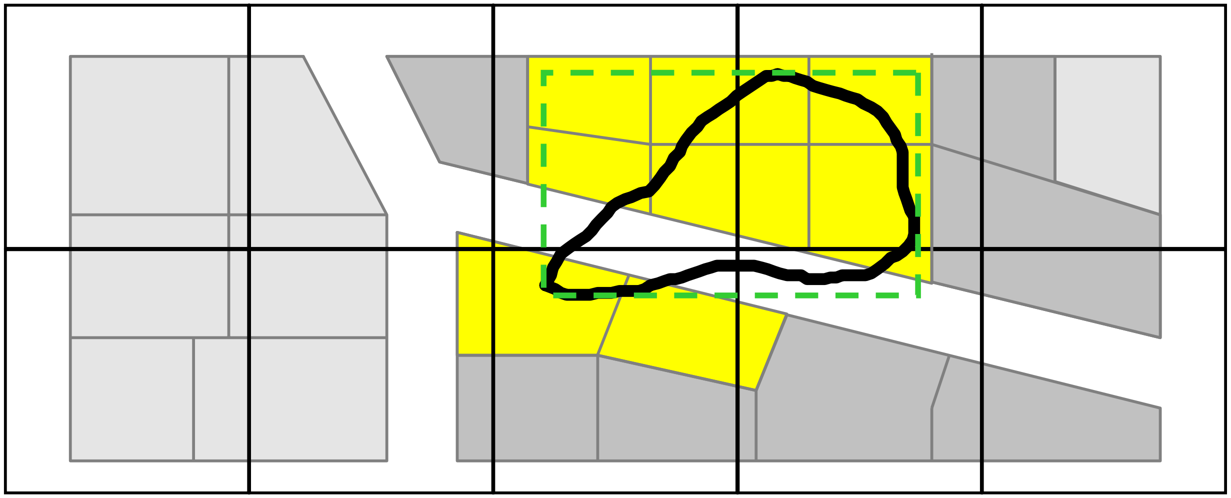
"retrieve all land parcels that overlap a ruptured storage tank’s contamination plume"



- Server determines which parcels share an index grid with the plume

Simple Envelope Test

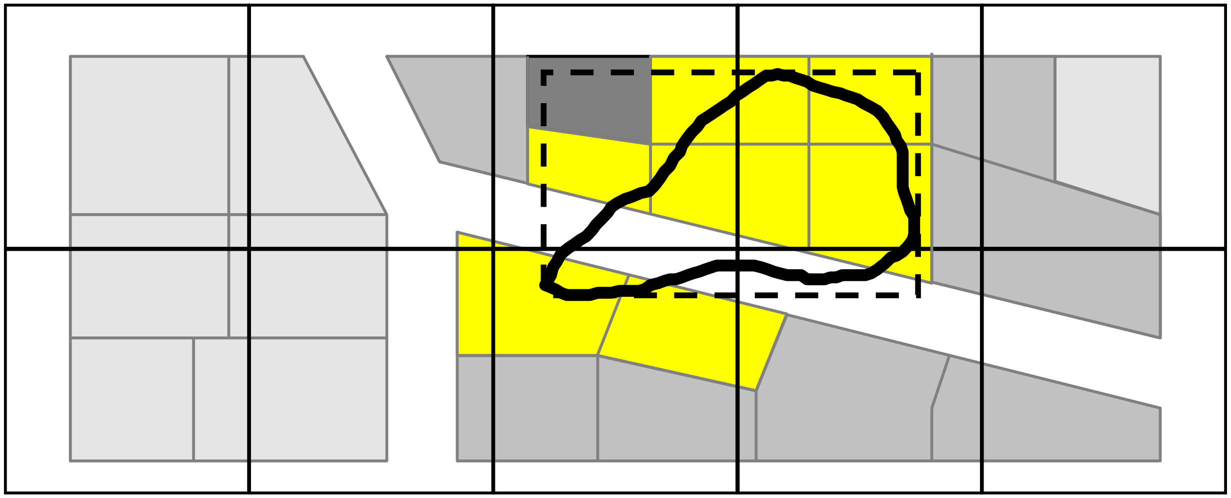
"retrieve all land parcels that overlap a ruptured storage tank's contamination plume"



- Server finds shapes with overlapping envelopes

Topology Engine Compares the Rest

"retrieve all land parcels that overlap a ruptured storage tank's contamination plume"



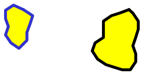
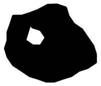
Topology Engine will accept or reject features based on complex topology comparisons

Business and Feature Tables

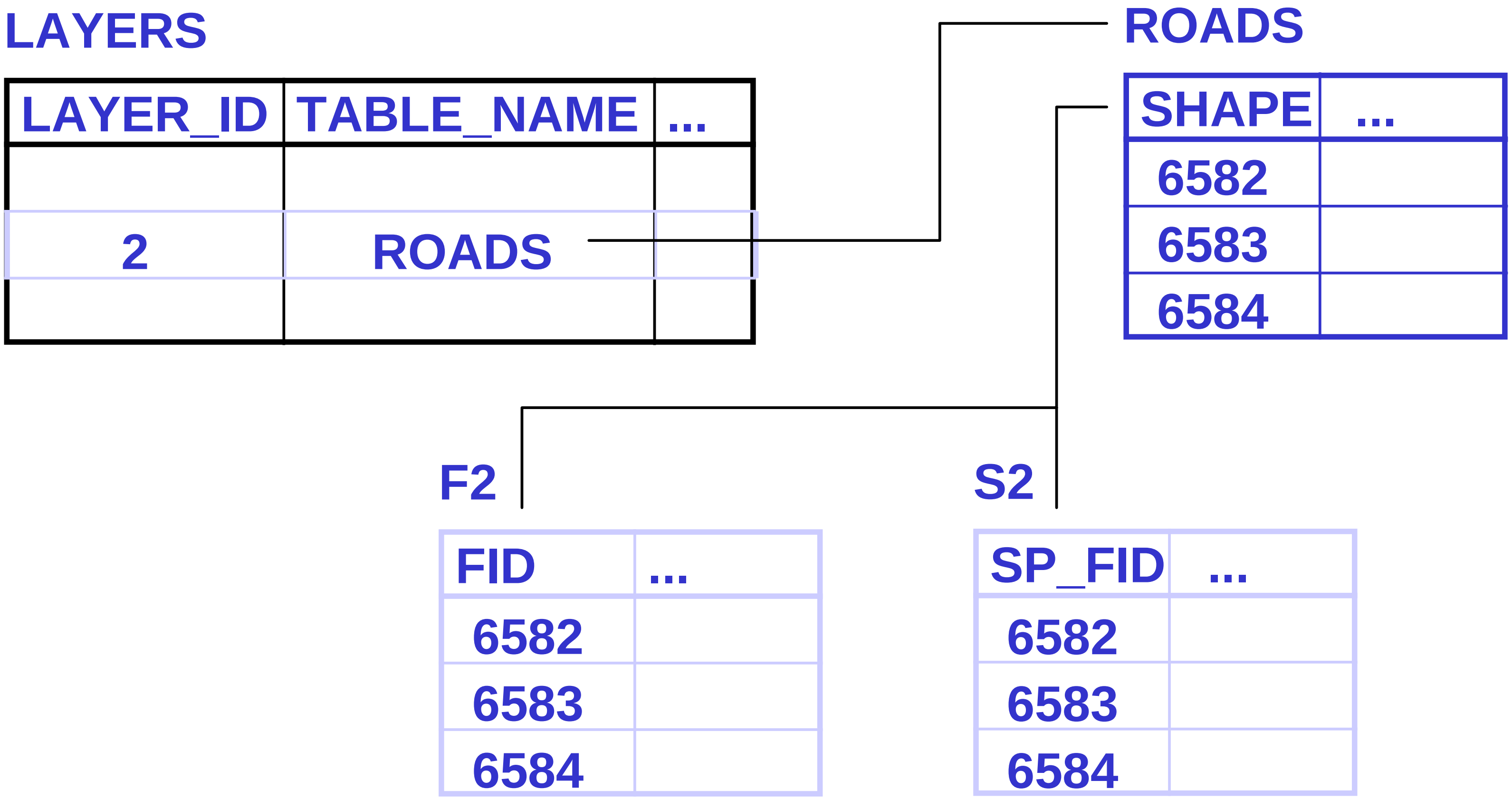
WILDERNESS AREAS

Name	Boundary
Big Flat	8984
Upper Valley	2170
East Ridge	3642

F9

FID	Points
8984	
2170	
3642	

Overview of SDE RDBMS Tables and Their Uses



LAYERS

LAYER_ID	TABLE_NAME	...
2	ROADS	

ROADS

SHAPE	...
6582	
6583	
6584	

F2

FID	...
6582	
6583	
6584	

S2

SP_FID	...
6582	
6583	
6584	

Inside the S Table

S Table Columns:

SP_FID, GX, GY, EMINX, EMINY, EMAXX, EMAXY

SP_FID is the join key.

GX and GY are grid coordinates used in the S table search.

The Envelope values are for trivial rejection of disjoint features. This avoids the fetch from the F table.

S Table Indexing

SDE 3.0

S<n>_IX1 on (GX, GY)

S<n>_IX2 on (SP_FID)

SDE 3.0.1

S<n>_IX1 on (GX, GY, and all of the rest)

S<n>_IX2 NOT MAINTAINED!

SDE 3.0.2

S<n>_IX1 on (GX, GY, and all of the rest)

S<n>_IX2 on (SP_FID)

Index columns that AREN'T in the where clause??!!

Normally one only indexes the columns that are queried. (i.e. in the where clause)

The RDBMS is smart enough to take the values from the index without going to the table IF the values are in the index.

The S table index in 3.0.2 takes advantage of this by indexing ALL the columns.

Now you don't have to worry about where you put the S Table. It is never hit.

Learn from this idea

Indexing both the “where” columns AND the “select” columns helps Business Table Queries in the same way that that it helped the S table queries.

Use this to help speed your attribute queries.

Index BOTH spatial column and your favorite column.

Remember: Indexes are ORDER DEPENDENT!!!

Grid Recommendations

- There are 3 Levels of grid.
- Do Not Use more than you need.
- This causes extra S table searches
- Most layers only need one grid level.

Grid Recommendations

First Level Grid should be set to the LARGER of:

- Your layers average feature size.

or

- Your average Querying Area.

Never make grids smaller than your features

2nd & 3rd Level Grids

Higher level grids should be used to “catch” features that are LARGER than your 1st LEVEL grid.

In most cases, if 80% of your features fit into your first grid, that’s good enough. Stay with one grid level.

Grids: Final Remarks

Even though Grids are tuned to an average query, there is almost always a “sweet spot” setting that works well for ANY query.

Ballpark settings typically get you close enough.

Grid tuning is a Black Art. Perform bench marks.

Hardware Issues

Server Hardware Components:

- CPU’s
- Memory
- Disk

CPU’s

- Faster is better
- More is better
- Single user benchmarks benefit less from more than 2
- Multiple threads (Multi Users) Benefit far more from more CPU’s

Memory

- More is better
- Saves Page Swapping
- Good for reselects
- Will not help initial selections
- Benefits seen especially on multi user systems

Disks

Get MORE than enough!!! Really!

RAID is BAD

RAID is GOOD

SCSI? NFS? IDE?

Spread out your I/O

The BIG LOAD

Create your tablespaces.

One for each layer.

Avoid Fragmentation!

Get the size right or your life will be hell!

Take your time and calculate EVERYTHING FIRST!

WRONG!!!!!

THERE'S NEVER TIME TO DO IT RIGHT.
THERE'S ALWAYS TIME TO DO IT OVER.

Its always faster to load a layer TWICE than to load it RIGHT the first time.

JUST DO IT!!

Here's how.

Fragment and Size LATER

F_INIT	1000000	F_IX1_INIT	1000000
F_NEXT	1000000	F_IX1_NEXT	1000000
A_INIT	1000000	A_IX1_INIT	1000000
A_NEXT	1000000	A_IX1_NEXT	1000000
S_INIT	1000000	S_IX1_INIT	1000000
S_NEXT	1000000	S_IX1_NEXT	1000000
		S_IX2_INIT	1000000
		S_IX2_NEXT	1000000

First Giveaway: ll_num

```
unix> ll_num

lists   OWNER          TABLE_NAME           LAYER_ID
        -----------    --------------------  ---------------
        SDE            COUNTIES              1
        SDE            STREET06              2
        LEO            FOO                   3
```

Second Give Away: layer_get_size

```
layer_get_size      owner/passwd
                    layer_name
                    layer_number
                    keyword
                    multiplier
                    next_scale
```

```
unix> layer_get_size sde/sde counties 2 COUNTIES 1.0   0.1
```

Analyzes tables to calculate exact size.
Outputs a DBTUNE.SDE entry

A word on ANALYZING TABLES

```
SQL> analyze table foo calculate statistics;
```

DON'T!!!

If you are getting lousy point-and-click identify response or if your layer displays fine in ArcVIEW but CRAWLS if you symbolize, Someone may have ANALYZED your tables and indexes

```
SQL> analyze table foo delete statistics;
SQL> analyze index foo_ix delete statistics;
```

ORACLE TEMP and RBS

After your layer loads, will it go into NORMAL_IO?

Suggestions:

Be sure your TEMP and RBS's are big enough.

Check that your TEMP table space default storage is OK.

Leave the layer in NORMAL_IO BEFORE you load it.

Use a COMMIT frequency value.

SDE Throughput

Is it as fast as it can be?

How many features per second?

Single CPU desktop server: 250 - 400 fps

Two CPU ULTRA CLASS: 350 - 500 fps

Really Expensive MainFrame: 850 fps

Tuning Myths and Realities

If you do the right stuff:

De-fragment your data.
Spread out your I/O
Tune Oracle
Tune the SDE grids
Tune the GIOMGR.defs
Index Properly

You keep making SDE faster a little at a time.

Reality

SDE throughput is a multi-step process.

You’re only as fast as your slowest step.

Speeding up a faster step WON’T help.

Focus on the things that matter.

Tune what Matters

Things that make a BIG difference

- Proper Attribute Indexing
- Grid Sizing
- Application Logic

Things that may not make a big difference

- Fragmentation
- Spreading out your I/O
- Oracle fine tuning.

Settle the Rumor

Local Shapefiles faster than SDE?

Ya. You betcha!

Flat files faster than a
Client-Server-RDBMS-Row-Topology-Network-Fetch?

Airline reservations would be faster if they used local copies of the seat assignment.

Remember what you get!

Centralized Server
Transaction Control
Backups
Concurrent Editing
No NSF Mounts

One copy of the data
Relational Algebra
Concurrent Access
VERY LARGE Datasets
Superior Security

Changes Immediately Visible by Everyone

Tricks with the RDBMS

Use Triggers instead of complex programming.

- Date stamps
- History layers
- Propagate data into other tables

Use Views instead of tables.

- Lighten MO’s attribute load.
- UNION ALL views “glue” two layers together.

Use Synonyms.

Hacking the layers table
(at your own risk)

Useful columns in the layers table.

OWNER	If you are moving and copying tables.
TABLE_NAME	You can put views and synonyms here.
LAYER_ID	This specifies the F and S table names.

If you change the LAYERS table, shutdown and restart SDE

RAID 5, Mirroring and Backups

unRAIDed Disk Crash: System down, Restore Backup

RAID5 Disk Crash: System down, Disk gets rebuilt.

Mirrored Disk Crash: System stays up, Disk gets rebuilt.

With RAID/Mirroring you don’t need to backup! (wrong!)

If you have the source data you don’t need really need RAID

More on Backups

If you are dealing with LARGE data sets, don’t go down!

If you install RAID do the restore “fire drill”.

Don’t think of system crashes and restores as possible.

Don’t think of them as an eventuality.

Think of them as routine.

FF (SDE) = I + T

Clustering data for Super Turbo Fast Display.

Some loss of functionality.

Huge throughput increases.

Some symbolization capability.

Great for “Wallpaper” and more...

Clustered Layers Utility

Usage:

```
<SDE server machine> <SDE instance name>
<source_table,sp_col> <dest_table,sp_col>
<username> <password>
<POINTS | LINES | ISOLATED_POLYGONS |
 TESSELLATED_POLYGONS | POLYGONS>
<CREATE | USE_EXISTING_LAYER>
<tile_size> <quoted_where>
<tag_val>[<SDE database name>]
```

Demo and Questions

Thank you and Keep in touch!

- Leo