Brad Gellerstedt and Mark Harley

Evaluating SDELOB and SDEBINARY Storage Mechanisms in ArcSDE

Abstract

When using ArcSDE with an Oracle database, the default compressed binary storage mechanism for geometry types is SDEBINARY, which uses the LONG RAW datatype in Oracle. However, this geometry storage option has some limitations. For example, Oracle replication cannot handle LONG RAW datatypes. Additionally, Oracle may desupport the use of LONG RAW in some future release. An alternative is to use the SDELOB mechanism, which uses BLOB datatypes in Oracle. However, are there any performance implications for converting to SDELOB? What other considerations should be addressed? This paper presents the results of an evaluation, benchmarking the SDELOB and SDEBINARY types. Recommendations are presented on the suitability of each method for specific situations.

Overview

ArcSDE is often used with Oracle as the underlying database. When using ArcSDE with Oracle, three geometry storage mechanisms are available: SDEBINARY, SDELOB, and SDO_GEOMETRY (Oracle Spatial). SDEBINARY and SDELOB both use ESRI's compressed binary format, whereas SDO_GEOMETRY uses Oracle's own spatial data format. More specifically, SDEBINARY uses the Oracle LONG RAW datatype to store the geometry, whereas SDELOB uses BLOB datatypes. Although SDO_GEOMETRY may be desired in certain situations, many ArcSDE users focus on the two compressed binary storage types: SDEBINARY and SDELOB. This paper will take a look at the reasons for using one or the other, as well as presenting some benchmarks which were performed to evaluate the performance implications of each.

ArcSDE Compressed Binary Storage Mechanisms for Oracle

SDEBINARY (LONG RAW)

The SDEBINARY storage type is probably the most widely used. It has been available since SDE version

3.0 and is still the default mechanism today. This storage mechanism uses a separate feature table (the "F" table) which contains, among other items, a single Oracle LONG RAW value that holds all the geometry for the feature (see Table 1). SDEBINARY is a very fast and efficient method for storing and retrieving spatial data.

SDEBINARY and SDELOB "F" Table Contents

SDEBINARY		SDELOB	
Column Name	*Column Type*	*Column Name*	*Column Type*
FID	NUMBER(38)	FID	NUMBER(38)
NUMOFPTS	NUMBER(38)	NUMOFPTS	NUMBER(38)
ENTITY	NUMBER(38)	ENTITY	NUMBER(38)
EMINX	FLOAT(64)	EMINX	FLOAT(64)
EMINY	FLOAT(64)	EMINY	FLOAT(64)
EMAXX	FLOAT(64)	EMAXX	FLOAT(64)
EMAXY	FLOAT(64)	EMAXY	FLOAT(64)
MIN_MEASURE	FLOAT(64)	MIN_MEASURE	FLOAT(64)
MAX_MEASURE	FLOAT(64)	MAX_MEASURE	FLOAT(64)
AREA	FLOAT(64)	AREA	FLOAT(64)
LEN	FLOAT(64)	LEN	FLOAT(64)
POINTS	LONG RAW	POINTS	BLOB

Table 1

SDELOB (BLOB)

The SDELOB storage type has been available as an option since ArcGIS 8.1. It is similar to the SDEBINARY type in that it stores the geometry in the "F" table. The SDELOB "F" table is, in fact, identical to SDEBINARY except that the value that stores the geometry is an Oracle BLOB datatype rather than LONG RAW (see Table 1). It is also a fairly efficient storage mechanism, but, as this paper will demonstrate, the BLOB datatype can be slower than LONG RAW in some cases.

There are reasons why SDELOB may be considered instead of SDEBINARY. These primarily involve Oracle's differing level of support for the BLOB and LONG RAW datatypes. In particular, some advanced Oracle features do not support LONG RAWs, but do support BLOBs. The primary feature of interest is

replication. This technique allows database objects (such as tables) to be replicated among multiple instances in a distributed database system. Oracle Advanced Replication and read-only snapshot replication are both supported by ArcSDE – if the SDELOB datatype is exclusively used. Therefore, if replication is desired, SDELOB must be used. Also, Oracle has stated that LONG RAW is a "deprecated" datatype and will be desupported at some future release; obviously, when that should come to pass, the decision to use SDELOB would become automatic.

Performance Benchmarks

One of the primary things to consider when choosing a storage mechanism is of course how well it performs. The rest of this paper deals with a set of benchmarks that was performed to determine how the SDELOB storage type measures up to SDEBINARY in terms of data retrieval, specifically based on drawing times. These tests were performed to see the "real-world" behavior of the two options for a number of layers of interest. The tests focus on vector data only (i.e., non-raster) - two-dimensional points, lines, and polygons.

Testing Environment

The benchmarking examples presented in this paper were performed using the following computing environment. The server hardware was an HP C5600, with dual 550mhz PA/RISC CPUs, and 2gb of RAM. Running on the server was ArcSDE 8.3 using Oracle 9i Release 2 as the relational database and an HP-UX 11.0 operating system. The client hardware was a Dell Latitude C600 with a 1ghz processor and 512mb of RAM. The client software was ArcInfo version 8.3 running on a Windows 2000 system. The tests used a variety of datasets based on GDT's Dynamap/2000 product; data was loaded for the entire state of New York for most tests. Testing was performed using ArcMap to render a map at a variety of different scales, depending upon the type of data being displayed. Emphasis was placed on retrieving large quantities of data (often full-table scans) to see how the two storage mechanisms truly measured up. Display times were determined using a simple timer control, developed in Visual Basic using ArcObjects. Tests were run a minimum of five times in succession to determine minimum and maximum display times; a mean display time was then determined from these values. In most instances, the times were extremely consistent.

Line Features

Linear features were investigated the most thoroughly, due to their importance in a street centerline database. The Dynamap/2000 Street layer was evaluated by loading data for the state of New York and using ArcMap to draw maps at a number of different scales, focused on several different parts of the state. The differences in draw time between SDELOB and SDEBINARY were reasonably consistent in these tests. Figure 1 shows the mean draw times for a map centered on Buffalo, NY, at four different scales.

Mean Draw Times for Street Layer (centered on Buffalo, NY)

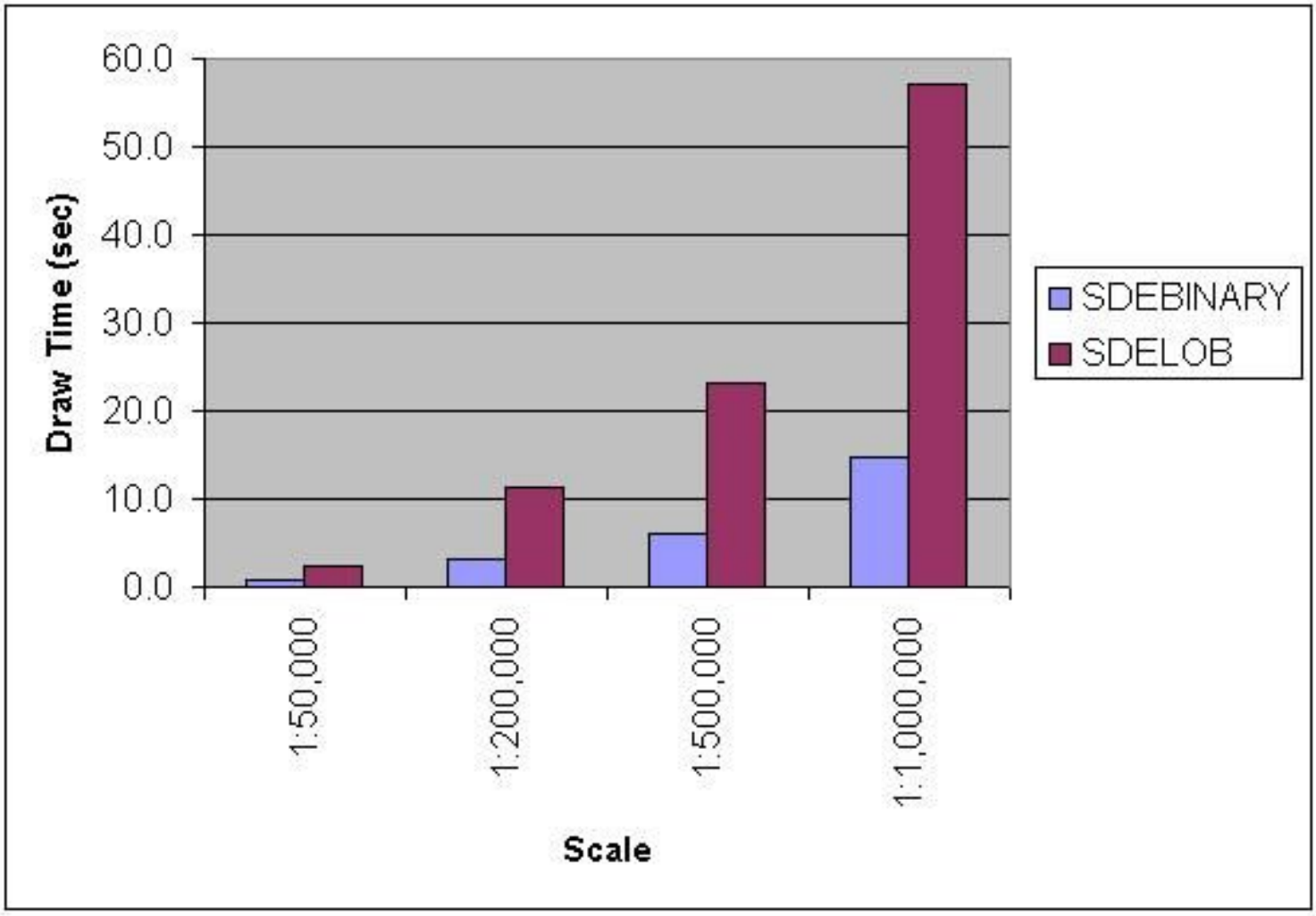

Figure 1

Figure 2 shows the ratio of these draw times between SDELOB and SDEBINARY. For the Street layer, it appears that using the SDELOB storage mechanism results in draw-time performance that is 3 to 4 times as long as for SDEBINARY.

Ratio of Draw Times for SDELOB to SDEBINARY for Street Layer

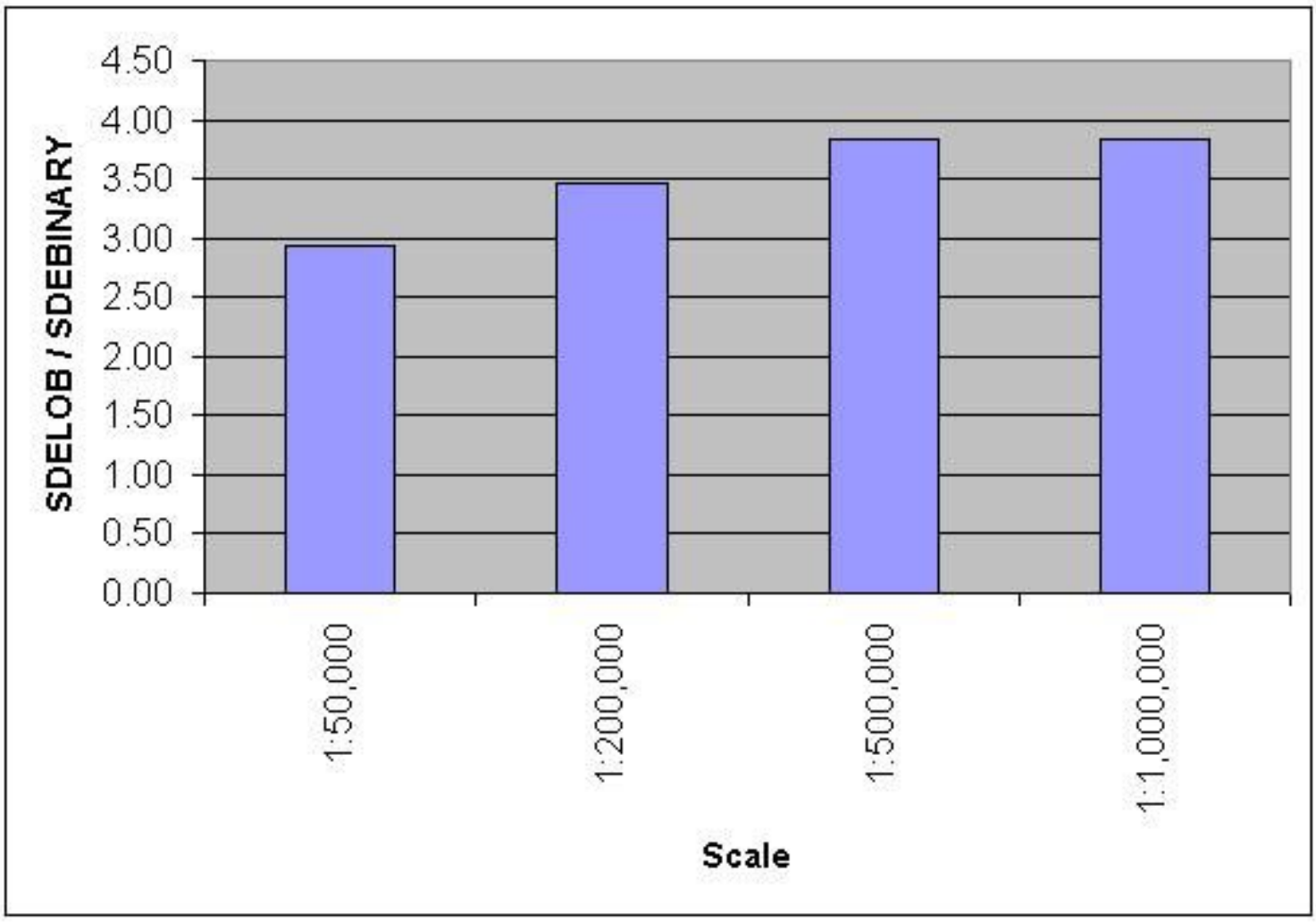

Figure 2

Other linear layers were also investigated. Most of these tests were focused on a single scale which typically displayed a majority of the features in the layer. Most of these layers were loaded just for the state of New York; however, to get a significant number of features to test, the Interstate layer was loaded for the entire United States. In addition, a multi-part feature layer, created using a technique similar to the sdegroup command, was created for the Street layer. This layer contains the same set of geometry as the Dynamap/2000 Street layer, but stores them in significantly fewer rows in the business and "F" tables; however, it lacks the detailed attribute information for each individual street segment.

The SDELOB-to-SDEBINARY draw-time ratios for these other linear layers are given in Figure 3. Note that the draw-time ratios vary from as high as 2.5 down to very nearly 1.0, which would represent essentially identical performance for the two storage mechanisms.

Draw-Time Ratios for Various Linear Layers

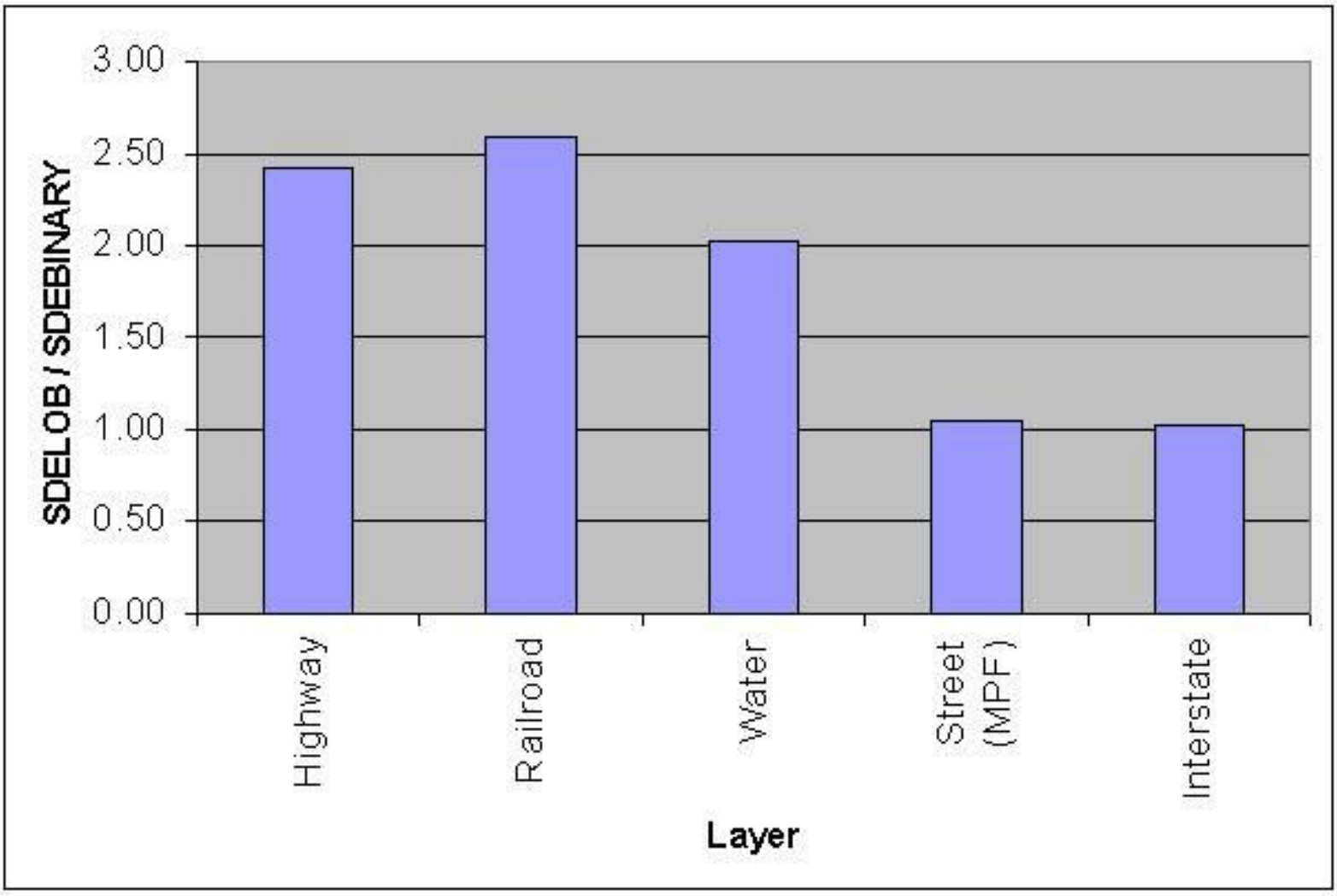

Figure 3

Polygon Features

Polygonal features were also investigated to see how the draw times for SDELOB and SDEBINARY stack up. A number of layers were tested, some for the state of New York (Tract, Block Group, Water Polygon, and Park), some for the whole U.S. (the Postal and County Boundary layers), and one (Block) for a portion of California. The draw-time ratios for these layers are given in Figure 4.

Draw-Time Ratios for Various Polygon Layers

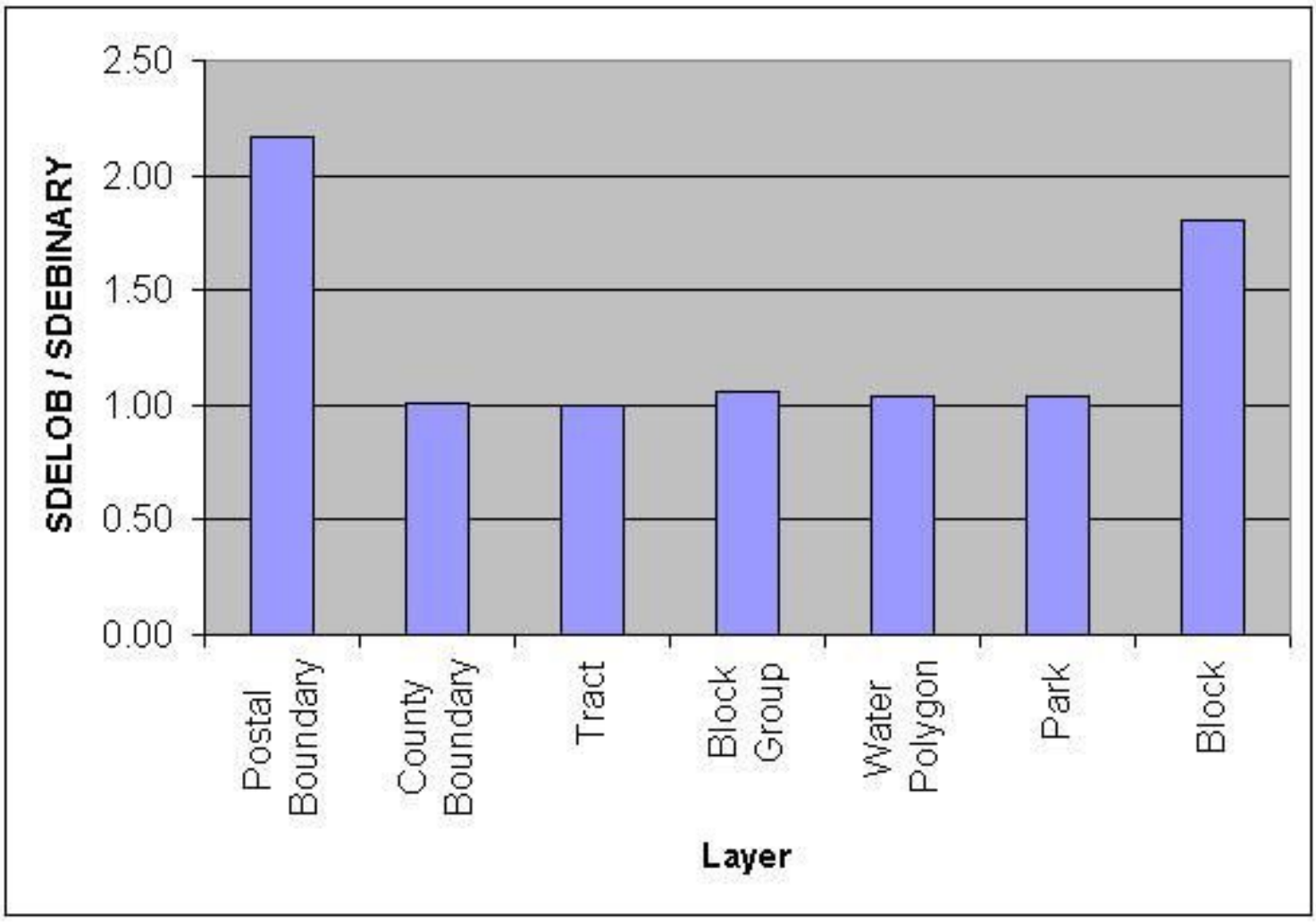

Figure 4

Note that all but two of these layers (Postal Boundary and Block) have a draw-time ratio of nearly 1.0. Apparently, at least these polygonal layers perform much better using SDELOB than many of the linear layers. A look at why this may be the case follows in the ***Correlations*** section.

Point Features

Only two point layers were investigated, Institution and Recreation Area. These two layers resulted in draw-time ratios of 3.92 and 2.33 respectively. The Institution layer is much more densely populated (300,000+ features for the U.S.) than the Recreation Area layer (less than 5,000 features) and, therefore, may represent a more complete test case.

Correlations

It has been noted that many polygon layers as well as some of the linear layers with large features (e.g., Interstate and the Street MPF layer) performed significantly better using SDELOB than the other layers. This suggests a potential correlation between the average "size" of the individual features and the draw-time performance with SDELOB. This indeed appears to be the case. Figure 5 displays a scatter plot of all the tests mentioned in the previous section, showing the draw-time ratio as a function of the average number of points-per-feature in the layer. The average points-per-feature was determined using the ***sdelayer –o stats*** command.

Correlation between Average Points-per-Feature and Draw-Time Ratio

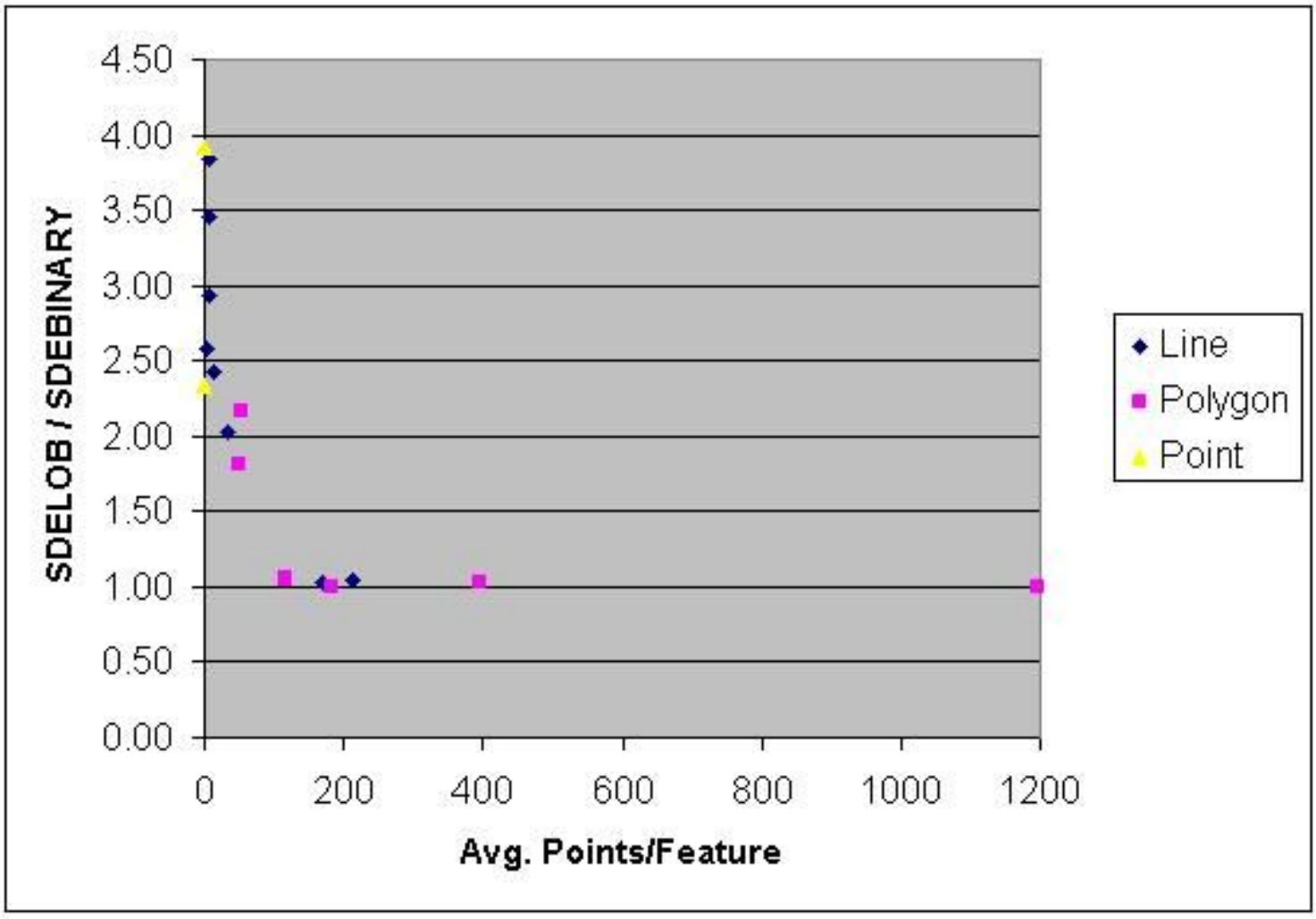

Figure 5

If the two layers with the most points-per-feature (Park and County Boundary) are removed for the sake of

clarity, a curve even begins to suggest itself (Figure 6). It appears that as the number of points-per-feature approaches approximately 100, the performance of SDELOB approaches that of SDEBINARY. However, as the number of points-per-feature approaches one, the SDELOB performance suffers greatly. This suggests that the complexity of the features has a great impact on the draw-time performance of SDELOB when compared to SDEBINARY – namely, the more complex the feature, the more similar the performance between the two storage mechanisms. Simpler features (such as points and "small" line features), on the other hand, are fairly inefficient when stored using SDELOB.

Correlation between Average Points-per-Feature and Draw-Time Ratio (with 2 extreme points removed)

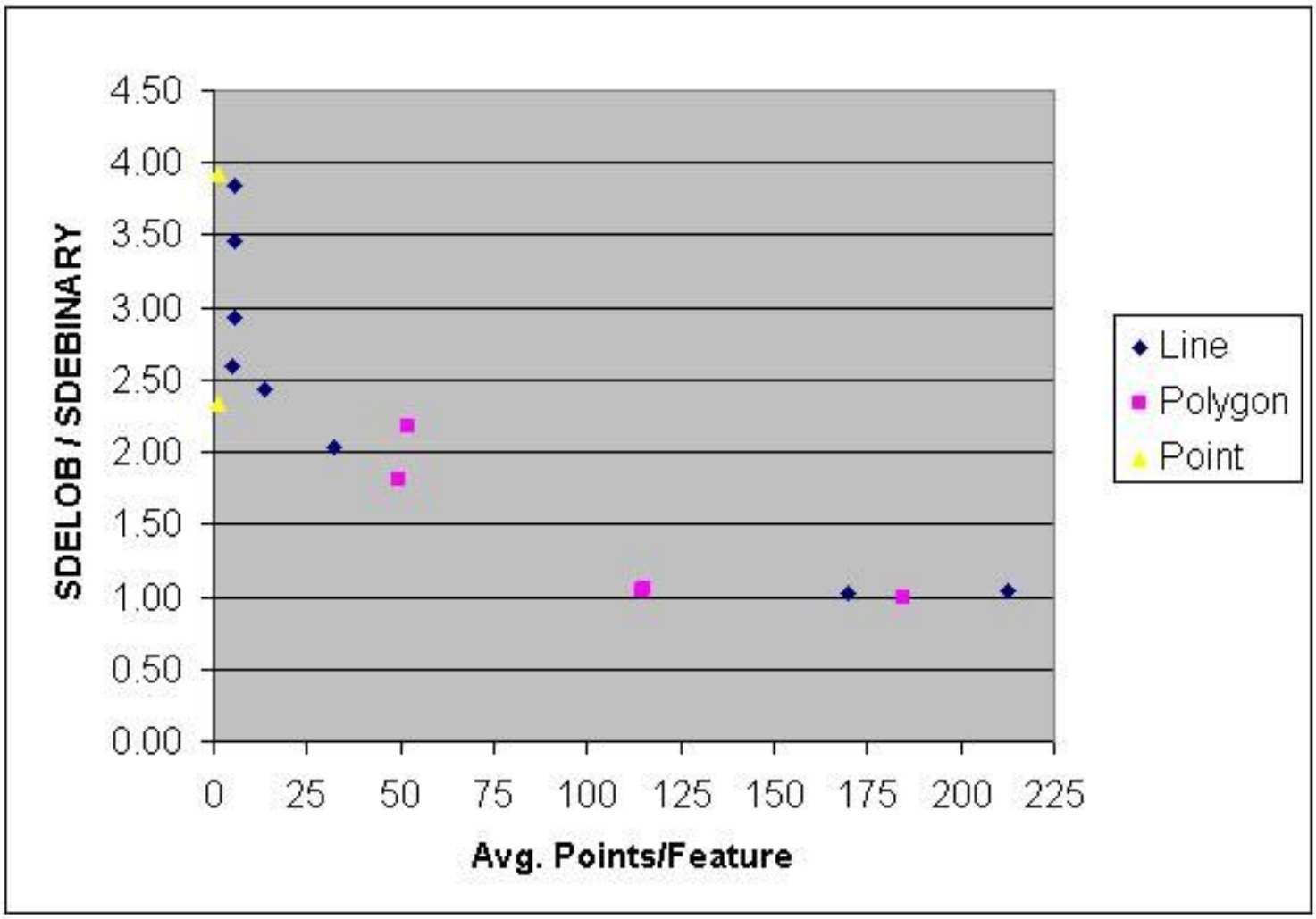

Figure 6

A similar correlation was revealed when looking at the total size of the "F" table along with any associated

BLOB segments. Oracle stores BLOB data in-place unless its size exceeds approximately 4000 bytes; BLOB values larger than this size are pushed out to a separate segment. For the tests performed here, several of the polygon layers as well as the Street multi-part feature layer contain a significant amount of data that exceeds this limit. However, as these layers all perform well with SDELOB, the fact that BLOB data has to be retrieved out-of-line does not appear to have a negative effect on draw time. Oracle does provide a column-level parameter called {ENABLE | DISABLE} STORAGE IN ROW that controls whether or not to place all BLOB values into the separate LOB segment. The default is ENABLE STORAGE IN ROW, which stores the value in the row as long as its length is less than 4000 bytes. Because this parameter cannot be changed once the table is created, it is always enabled for the "F" table, managed directly by ArcSDE.

Perhaps not surprisingly, the slower performing SDELOB layers have a larger "F" table (plus BLOB segment) size than their SDEBINARY counterparts. Figure 7 shows the correlation between this "F" table "size ratio" and the draw-time ratio. The "size ratio" represents the total number of blocks used for the "F" table for the SDELOB layers plus the total number of blocks for any associated BLOB segments with respect to the size of the "F" table for SDEBINARY. There does appear some correlation here, although it is not strictly 1:1. In fact, the draw-time performance appears to be quite sensitive to this total "F" table size. This should all be somewhat intuitive – if more blocks are touched when retrieving the data, the draw-time increases.

"F" Table Size Ratio vs. Draw-Time Ratio

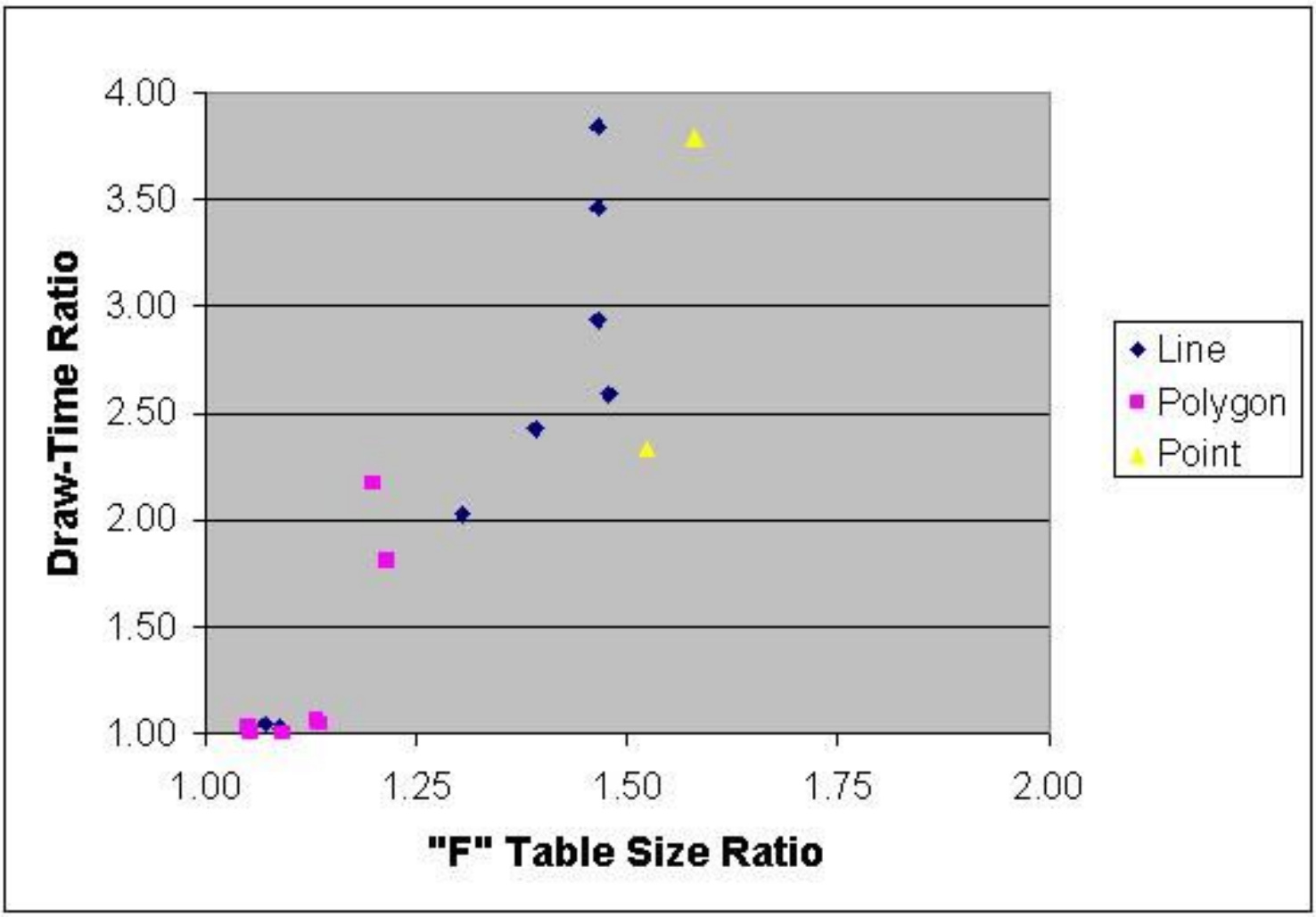

Figure 7

Conclusions

This paper has shown that, at least for a particular set of vector layers, drawing performance in ArcMap is greatly affected by the geometry storage type used in ArcSDE with Oracle. Specifically, the SDELOB type performs significantly slower than SDEBINARY, except when the average number of points per feature is high, as in large polygons. Additional investigation may be necessary to determine if any Oracle and/or ArcSDE tuning can be performed to improve on this behavior. Barring any significant tuning discoveries, it seems that unless the need for Oracle replication is great, it would be prudent to continue to use SDEBINARY for the type of data investigated in this paper.

Be advised, however, that these results are specific to a particular set of layers, on a particular version of ArcSDE, running on a particular database instance, accessed from a particular client environment. Results may indeed be very different for other types of data in other environments.

AUTHORS

Brad Gellerstedt, Senior Software Engineer
Mark Harley, Director, Consultative Services
Geographic Data Technology, Inc.
11 Lafayette St.
Lebanon, NH 03766-1445
Telephone: (603) 643-0330
Fax: (603) 653-0249
E-mail: brad_gellerstedt@gdt1.com, mark_harley@gdt1.com
Web: http://www.geographic.com