Geodatabase Precision

Document prepared by Beth Hill, ESRI-Denver

Geodatabase precision has a very different meaning than coordinate precision for coverages. Geodatabase precision is the "conversion factor" between your floating-point map units and your integer storage units.

precision = map units / desired storage units
Data is in State Plane Feet
Map Units = Feet
Desired storage accuracy: ¼ inch
Precision = 1 Foot / ¼ inch
Minimum Precision = 12 inches / .25 inches = **48**

See exercise 3B from the Building Geodatabase class for how to calculate the minimum precision necessary to maintain your coordinates.

How the default precision is calculated:

ArcCatalog will look at your map units and calculate the largest precision value possible to fit within the maximum of ~2.14 billion storage units depending on the coordinate extent (min/max X and Y)

Effects of Precision:

Too high:

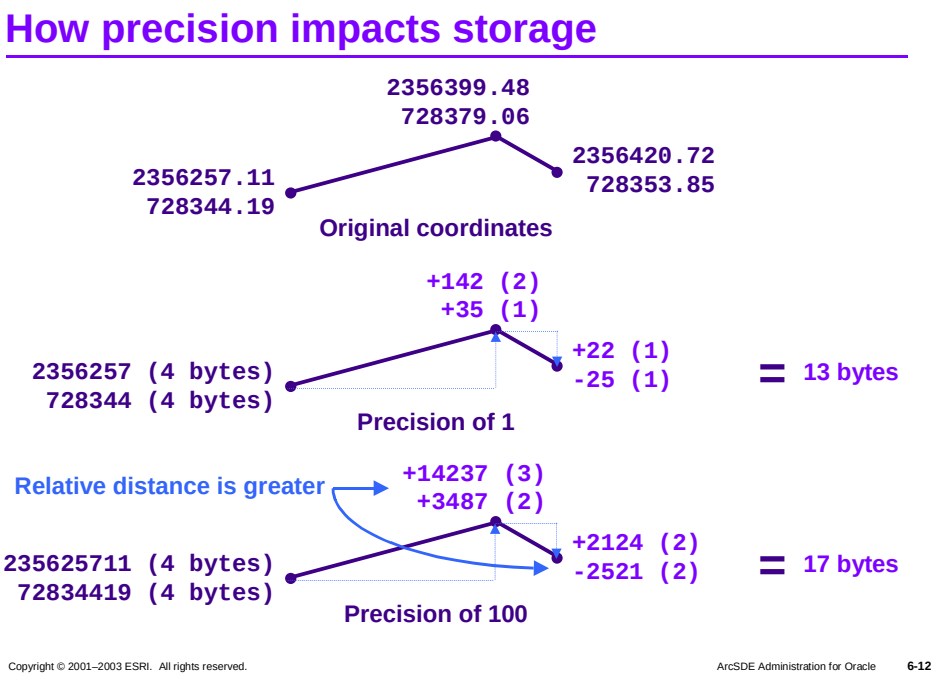
- **Small performance impact** (possibly only milliseconds) due to additional numbers being stored and more space needed to store them. (See graphics below)
- **Over-inflation of accuracy**- you will see far more decimal places than your data is accurate to, giving a false impression of the accuracy of the data
- Limit to how large you could set your topology **cluster tolerance** to

Too low:

- **Shifting of features** due to a loss in accuracy. Decimal places will get dropped, coordinates will get rounded, data can shift and become generalized
- Limit to how small you can set your topology **cluster tolerance** to.

(For topology, minimum cluster tolerance is approx. 2/precision. Max. cluster tolerance is approx. minimum * 100,000)

From the SDE for Oracle class:



"When stored with a precision of 1, the geometry only requires 13 bytes of storage space. With a precision of 100, however, the same geometry requires 17 bytes of storage. In the latter example, more information from the original floating-point numbers is preserved so the higher-resolution geometry demands more storage."

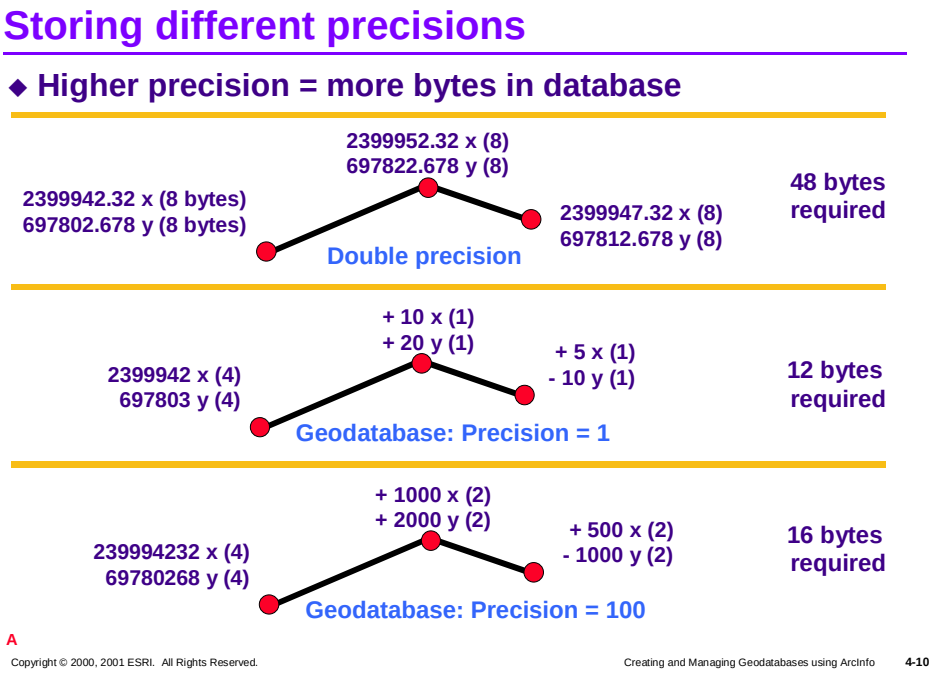
"There exists a direct relationship between a feature class's precision and the storage space required by its `POINTS` column. With the `WILSON.PARCELS` feature class, for example, if you accepted the default precision of 3,906 calculated by ArcCatalog, your coordinates would require 71 percent more space in the database than they do with your realistic precision value of 12.

Choosing an appropriate precision value instead of accepting the defaults calculated by your loading tools is one of the easiest and most substantial ways to improve the performance of your ArcSDE geodatabase. Reducing the amount of data stored reduces all forms of I/O, including critical disk and network transfers, and makes more efficient use of valuable physical RAM in the block buffer and log buffer."

From the old Creating and Managing Geodatabase class:

"**Coordinate storage performance tips**

Earlier in this lesson you learned that the ArcInfo coverage data model provides two ways to store coordinates (single or double precision). Single precision provides six or seven precise digits, and if you need to store higher quality coverage data at small or large areas, you can use double precision, which offers 13 or 14 precise digits. You also learned that the geodatabase model provides zero to ten precise digits, depending on your precision multiplier.



In the above example, a line stored in double precision requires eight bytes per coordinate. A total of 16 bytes is needed to store each endpoint or vertex because they each require two coordinates (x and y). That means that a simple three-point line requires a total of 48 bytes of coordinate storage.

The geodatabase has significant advantages to how the numbers are stored. For instance, subsequent vertices along a line are stored with their *differences* from the previous vertex. This means that the starting coordinate will always require 4 bytes but subsequent coordinates will be stored with between 1 and 4 bytes (depending on the distance between coordinates). When you increase the precision, the distance between the vertices is greater, therefore you may need more bytes to store these subsequent coordinates. Given the fact that higher precision requires more storage space, you should carefully consider how much precision you really need. "