HEXAGON GEOSPATIAL

GeoMedia 2016
Oracle Spatial Data Server
User Guide

June 2016

CONTENTS

ORACLE CONNECTIONS

GeoMedia provides an Oracle Object Model data server that facilitates connections to Oracle based warehouses. This allows GeoMedia applications to access both Oracle simple data types and Oracle location-based data in SDO_GEOMETRY format. There are two variations of the Oracle Object Model Data Server, a read-only data server and a read-write data server. These are accessed through the **New Connection** command. This document applies to both variations.

Prerequisites

GeoMedia will automatically install the Object Model Data Server. Registration of this data server requires that Oracle 32 bit client software (insta-client, runtime or administrative) be present on the system before installing GeoMedia. If the Oracle client has not been installed, the Oracle connection types will not appear in the Connection types list on the **New Connection** dialog box in GeoMedia Professional. Oracle 11g is the earliest client version that is supported (11.2.0.3 or later). If you have installed GeoMedia before installing the Oracle client, you will need to run the Configuration Wizard once the Oracle client has been loaded.

The Object Model Data Server requires the standard INTERMEDIA LOCATOR package in Oracle. The Oracle spatial package is not required (unless you are using GeoRaster capability) but is fully supported. ORACLE SPATIAL is a superset of INTERMEDIA LOCATOR and adds commands for complex spatial analysis, LRS, and other advanced capabilities. The Object Model Data Server is fully compatible with both Spatial and Locator. Refer to the ORACLE SPATIAL USER'S GUIDE and REFERENCE documentation for information on storing spatial data in Oracle and on the difference between Locator and Spatial.

Read-Only Connections

To make a read-only connection to an Oracle warehouse, you must provide a valid Oracle connection string, which usually consists of an Oracle username, password, and net service name. Net service names are created using Oracle's Net Configuration Assistant and reflect the database you are trying to connect to (the information is stored in the TNSNAMES.ORA file). Oracle's Easy Config connect strings(user/pswd@server/sid) are also supported. Read-only connections can be made on Oracle's native data model; no specific GeoMedia metadata information is required. There will be limits on what GeoMedia can interpret from existing spatial data, but metadata itself is not a requirement for display. In fact, you can use the Oracle Object Model data server to view and to use any tables within the GeoMedia environment as long as the connection is read-only.

If you are working without metadata, the GeoWorkspace must be assigned a coordinate system that matches the spatial data that is being displayed. This ensures that the data is displayed in the correct spatial location. GeoMedia will interpret all geometry types as Compound. For best results with data access and performance, you should always configure GeoMedia's metadata. If GeoMedia's metadata is detected, all read-only connections will require metadata entries.

Read-Write Connections

A read-write connection to an Oracle warehouse requires the same connection parameters as the read-only connection. However, read-write connections also require a metadata schema to be present

in the Oracle database instance. The user for this schema must be called GDOSYS, and it can reside on any Oracle tablespace (20M expandable to 50M is usually sufficient). For best results and for performance reasons, assign GDOSYS to its own tablespace. The metadata tables in GDOSYS store information used by all the schemas that require read-write access from GeoMedia applications. The GDOSYS schema is also used to store coordinate system information for all feature classes.

During the connection to Oracle, the Oracle data server will scan all accessible database objects. If the data server detects the existence of the GALIASTABLE table in the GDOSYS schema, metadata entries for all spatial tables, standard tables, and views will be required before GeoMedia Professional will display them.

FOR MORE INFORMATION, SEE "GEOMEDIA'S GDOSYS METADATA SCHEMA" IN THIS APPENDIX.

Password Persistence

By default, GeoMedia stores the Oracle connection password in the GeoWorkspace. This is meant as a convenience and allows users to open existing GeoWorkspaces containing Oracle connections without having to re-enter connection passwords. However, this is a drawback to those users wanting higher levels of security. The option to turn off password persistence is in the registry:

HKEY_LOCAL_MACHINE\SOFTWARE\GDO\ORACLE OBJECT READ-ONLY\1.0\STORE PASSWORD

HKEY_LOCAL_MACHINE\SOFTWARE\GDO\ORACLE OBJECT READ-WRITE\1.0\STORE PASSWORD

The default setting is 1, which means that connection passwords will be stored. To force the user to enter a password for each Oracle connection, change the (default) setting to 0.

Password persistence is not an issue if you are using Windows domain authentication for your connections.

Domain Authentication

Connections to Oracle based schemas can utilize either Oracle authentication (the default mode) or Windows domain authentication. To use Windows domain authentication, you need to first set some Oracle configuration parameters.

In the SPFILE or the initialization file, INIT.ORA, you need to set the following:

- REMOTE_OS_AUTHENT=TRUE
  this enables remote authentication in the instance.
- OS_AUTHENT_PREFIX=<auth_prefix>
  this sets the prefix Oracle will use for domain authenticated user names. GeoMedia does not support the default prefix OPS$. You will need to choose a prefix that does not contain the $ character. The only special character allowed here is the underbar(_).
  For example:
  REMOTE_OS_AUTHENT=TRUE
  OS_AUTHENT_PREFIX="DA_"
  You may need to restart the database instance after setting these values.

Create you user account in Oracle using the OS_AUTHENT_PREFIX, and specify EXTERNAL AUTHENTICATION for the password. For example, if your domain account is JSMITH, your Oracle user name is DA_JSMITH. As with normal Oracle user accounts, you need to assign the appropriate roles and privileges to this user, typically connect and resource, but that is up to the Database Administrator (DBA). To connect in GeoMedia, set the connection option to User Windows authentication, and enter the database service name.

If your domain authenticated username contains any special characters (such as **/** or **-**), you can still use it to connect, but it will not be able to own any database objects. GeoMedia uses an OWNER.TABLE syntax when working with tables/views and special characters will cause this to fail.

Object Model Data Server Requirements

The Oracle object model data server has specific requirements and limitations. These are listed below:

- Geometry is stored in a column of type SDO_GEOMETRY. Multiple geometry columns are allowed per feature class, but only one geometry field can be primary. GeoMedia will display all available geometry fields, but only the primary geometry will be editable. The primary geometry field can be changed through **Database Utilities**. Feature classes containing multiple geometries cannot be created using the GeoMedia Professional **Feature Class Definition** command.
- Only one set of attribute data is allowed for any given geometry. The use of attribute tables in a join-view relationship with a table containing geometry information allows you to get around this limitation.
- All spatial filter operations are performed on the Oracle Server, which requires spatial indexes to exist for all feature classes. This greatly improves spatial filter performance for filter areas that are less than 70% of the total area covered by the feature class.
- Spatial indexes require the presence of Oracle's spatial metadata in USER_SDO_GEOM_METADATA. GeoMedia requires entries in this metadata view for all tables and views containing geometries.
- Mixed-case table and column names are not allowed. All values representing database object must be in uppercase and must conform to the standard Oracle conventions for table names. Oracle has many reserved words and these must not be used for table or column names. Using reserved words can have unpredictable results.
- When creating feature classes using **Feature Class Definition**, table names are restricted to 24 characters. GeoMedia Professional reserves six characters for index/sequence names. The use of the $ character in table names is supported, but you should not use $ in column names. If you are planning to use Oracle's **Workspace Manager**, you are restricted to 22-character table names.
- If your tables are created directly in Oracle, you can use Oracle's limit of 30 characters per table and column name and any sequences and indexes must also be created manually. For Workspace Manager, the Oracle-imposed limit is 25.
- To be editable, all tables must have a primary key. If the table does not have a primary key, it will be read-only.
- Multiple column primary keys are allowed (up to 7), and they can have both a numeric or character datatype, or any combination thereof.

- Primary keys can be numeric or alphanumeric. Integer-based primary keys, populated by an associated sequence, are recommended and will provide the best results.
- For views, a primary key is required for at least one of the tables that will be used in the view definition. This is known as a key preserved view. The key column must be indicated to GeoMedia through the GDOSYS metadata table GINDEX_COLUMNS. In order to be insertable, GeoMedia must furnish the key value, either through direct key-in or through autonumber assigned sequence.

  FOR MORE INFORMATION ON THE FORMAT AND CONTENTS OF THIS TABLE, SEE "GINDEX_COLUMNS" IN THIS APPENDIX.
- The Oracle Object Model data server supports the use of sequences for each field that needs to be treated as AutoNumber. The most common use of AutoNumber is primary key fields. GeoMedia determines the existence of the sequence by looking in the GDOSYS metadata table GFIELDMAPPING. Use **Database Utilities** to create and/or assign sequences to the appropriate fields.

  FOR MORE INFORMATION ON THE FORMAT AND CONTENTS OF THIS TABLE, SEE "GFIELDMAPPING" IN THIS APPENDIX.

  NOTE These sequences will be used by inserts made within the GeoMedia environment. If you want to use the sequences for edits made outside of GeoMedia, you will need to set up an insert trigger. See "Triggers" in this appendix for more information.
- The Oracle Object Model Data Server is designed for multi-schema support. All Oracle DCL privileges are fully supported. Use the GRANT command in SQL to allow one user to connect to another user's schema. The minimum level of privilege required is SELECT. To facilitate this in GeoMedia, all table names are prefixed with the name of the schema from which the table came, for example, INGR.ROADS.
- The use of table and view synonyms is not supported at this time.
- Oracle spatial data that utilizes LRS geometry is supported in a read-only mode. You are allowed to view the associated geometry in GeoMedia Professional, but the measures are ignored. You must use Oracle views to directly access the measured values.
- All views that are updateable in Oracle are updateable in GeoMedia Professional. The GDOSYS metadata table GINDEX_COLUMNS needs an entry for the view that tells GeoMedia Professional what the primary key of the view's base table is. This can actually be any column in the view that behaves like a key column (unique and not null). You can also use multi-column primary keys here. If the view is not key preserved, it will be read-only.

  FOR MORE INFORMATION ON THE FORMAT AND CONTENTS OF THIS TABLE, SEE "GINDEX_COLUMNS" IN THIS APPENDIX.

GEOMETRY STORAGE AND TYPE MATCHING

Oracle Spatial/Locator uses a pre-defined database object type to store spatial data. The object type is defined in the database as MDSYS.SDO_GEOMETRY. The basic components of this data type are described as follows:

GEOMETRY_COLUMN (SDO_GTYPE, SDO_SRID, SDO_POINT (X,Y,Z), SDO_ELEM_INFO(OFFSET,ETYPE,INTERPRETATION), SDO_ORDINATES(X,Y,Z))

FOR MORE DETAILED, UP-TO-DATE INFORMATION, SEE THE ORACLE SPATIAL USERS GUIDE and REFERENCE.

- SDO_GTYPE – The GTYPE indicates the geometry type (point, line, area, and so forth) and the dimension (2-D or 3-D) of the entire feature class.
- SDO_SRID – The SRID is used to identify the native Oracle coordinate system (spatial reference system). GeoMedia Professional uses its own coordinate system indicator and ignores this field. The field is NULL by default but may contain values. In the case of geodetic data, an SRID is required by Oracle. Consult the ORACLE SPATIAL USERS GUIDE and REFERENCE document for more information on the SRIDs. If you plan to do any server-side analysis on spatial data, you should assign an SRID.
- SDO_POINT – For storing a single point in X, Y, Z. GeoMedia uses oriented point geometry and does not use SDO_POINT. It will read data written to SDO_POINT but will not write data there. There is no advantage in using the native point geometry versus using oriented point geometry. By default, this field will be NULL.
- SDO_ELEM_INFO – This field is a variable length array of type NUMBER (maximum size 1048576). The values that make up this array are composed of triplets that describe how the ordinates are stored in the SDO_ORDINATES array. Each triplet is interpreted as follows:
  - OFFSET - Indicates the offset within the SDO_ORDINATES array where the first ordinate for this element is stored.
  - ETYPE - Indicates the type of the individual element.
  - INTERPRETATION - Determines how the ETYPE value is interpreted.
- SDO_ORDINATES – This field is a variable-length Oracle array of type NUMBER (maximum size 1048576) that stores the coordinate values that make up a spatial object. The limit on the number of 2-D coordinate pairs (X,Y) is 524288. For 3-D data, the limit on the number of coordinate triplets (X,Y,Z) is 349525. This limit is fixed by Oracle and cannot be changed. This array is always used in conjunction with the SDO_ELEM_INFO array. The values in the array are ordered by dimension in X, Y, and Z.

  For example, a polygon whose boundary has four two-dimensional points is stored as:

  ```
  SDO_ORDINATE(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X1,Y1)
  ```

  If the points are three-dimensional, they are stored as:

```
SDO_ORDINATE(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4,X1,Y1,Z1}
```

Geometry Type Mapping

The Oracle Object Model Data Server maps GeoMedia's geometry types into the SDO_GEOMETRY types, and vice versa. The following table identifies the mapping used when converting from Oracle geometry types to GeoMedia geometry types. The symbol “d” indicates the dimension, d=2 for 2-D data and d=3 for 3-D data.

Oracle to GeoMedia Element Type Conversion				
Oracle Gtype	**Oracle Etype**	**Oracle Interpretation**	**Oracle Meaning**	**GeoMedia Geometry Type**
d001	0	6000	Application Defined	GeoMedia Point Rotation Matrix
d001	0	6001	Application Defined	GraphicText
d001	1	1	Point	SpatialPoint
d001	1	0	Point Orientation (10g or later)	Point Rotation Matrix
d002	2	1	Linear – straight line segments only	SpatialLine
d002	2	2	Linear – arcs only	SpatialLine
d002	4		Linear – straight line segments and arcs	SpatialLine
d003	0	6002	Application Defined	Coverage (Raster)
d003	3	1	Areal – straight line segments only	SpatialArea
d003	3	2	Areal – arcs only	SpatialArea
d003	3	3	Areal – rectangle	SpatialArea
d003	3	4	Areal – circle	SpatialArea
d003	4		Areal – straight line segments and arcs	SpatialArea
d004 … d007			Heterogeneous Collection	SpatialAny

The following table identifies the mapping used when converting from GeoMedia’s geometry types to Oracle geometry types (applies only to data changes through the read-write data server).

GeoMedia to Oracle Element Type Conversion				
GeoMedia Geometry Type	**Oracle Gtype**	**Oracle Etype**	**Oracle Interpretation**	**Oracle Meaning**
SpatialPoint	d001	1	1	Point
Orientation Matrix		0	6000	Application-defined
Orientation Matrix		1	0	10g Point Orientation
GraphicText	d001	0	6001	Application-defined
SpatialLine	d002	2	1	Polyline
Spatial Line (Arc)	d002	2	2	Linear – arcs only
Composite Polyline	d002	4	2	Linear – straight line segments and arcs
Boundary	d003	1003	-	Exterior Ring
Boundary	d003	2003	-	Interior Ring
Composite Polygon	d003	1005	-	Composite Exterior Ring
Composite Polygon	d003	2005	-	Composite Interior Ring
Raster Polygon	d003	0	6002	Application Defined

In versions of Oracle prior to 10g, GeoMedia uses its own format for oriented points. GeoMedia's oriented point format adds an application defined Etype of 0 and a custom GeoMedia Interpretation of 6000 in the ELEM_INFO_ARRAY prior to the definition of the point. The corresponding entry in the ordinates array will contain the orientation of the point. An example of the SDO_ELEM_INFO_ARRARY containing a GeoMedia oriented point is shown below:

```
SDO_ELEM_INFO_ARRAY(1, 0, 6000, 4, 1, 1)
```

Oracle 10g introduces a native oriented point format. GeoMedia will automatically use this format if it detects 10g. In a native oriented point, the rotation matrix uses an interpretation of 0 and follows the point definition. An example of the SDO_ELEM_INFO_ARRARY containing a native oriented point is shown below:

```
SDO_ELEM_INFO_ARRAY(1, 1, 1, 3, 1, 0)
```

For polygon ring elements, 4-digit ETYPE values are required with the first digit indicating exterior (1) or interior (2). Ordering the storage of ordinates is also very important. The basic Eytypes for polygons are as follows:

- Exterior polygon ring – ETYPE=1003 and ordinates must be specified in counterclockwise order.

- Interior polygon ring – ETYPE=2003 and ordinates must be specified in clockwise order.

ETYPE values 4 and 5 are compound elements. They contain at least one header triplet with a series of triplet values that belong to the compound element. ETYPE 4 is a compound line string, and the following two 4-digit types represent compound polygons:

- 1005: exterior polygon ring (ordinates must be specified in counterclockwise order).
- 2005: interior polygon ring (ordinates must be specified in clockwise order).
- The INTERPRETATION value takes on a different meaning depending on whether or not the ETYPE is a compound element:
  - If the ETYPE is a compound element (#005), this field specifies how many subsequent triplet values are needed to define the element.
  - If the ETYPE is not a compound element (1, 2, or 3), the interpretation attribute determines how the sequence of ordinates for this element is interpreted. For example, a line string may be made up of a sequence of connected straight-line segments or circular arcs.

Several geometry types are not completely supported by GeoMedia. Both native SDO_POINT geometries and point clusters can be read and displayed, but GeoMedia will not write them. The same is true for arc-strings, native circles, and simple 2-point polygons.

Oriented Points

Simple point features do not convey much information other than a location on a map. Adding symbols or fonts to a point adds specific meaning and in most cases, these symbols and fonts have specific orientations. All point features in GeoMedia are symbolized oriented points rather than simple points.

Oracle's simple point format utilizes the SDO_POINT array in the SDO_GEOMETRY object, similar to the following example:

```
SDO_GEOMETRY (3001, NULL, SDO_POINT(861906,-482832, 0),NULL,NULL)
```

GeoMedia can read this format but will not write to it. Instead, GeoMedia uses Oracle's oriented point format shown below:

```
SDO_GEOMETRY(3001, NULL, NULL, SDO_ELEM_INFO_ARRAY(1,1,1,4,1,0),
             SDO_ORDINATE_ARRAY(861906,-482832,0, 0,0,1))
```

For 2-D points, only a 2-D matrix is required. Here is an example:

```
SDO_GEOMETRY(2001, NULL, NULL, SDO_ELEM_INFO_ARRAY(1,1,1,3,1,0),
             SDO_ORDINATE_ARRAY(861906,-482832,0,1))
```

Rotation matrices are used to improve the accuracy of calculations. A standard 2-D rotation matrix is i,j and to convert this to radians or degrees, you would use the following:

```
radians=ATAN2(j/i);
degrees=ATAN2(j/i)*(180/PI);
```

Text and Labels

Oracle currently does not have a native storage format for text. In order to make use of Oracle's existing format, GeoMedia stores text as a point feature with some additional information that contains text orientation, alignment, format, and the actual text decomposed into four-byte integers. An example of how GeoMedia stores the text CONNECTICUT in a SDO_GEOMETRY object is shown below:

```
SDO_GEOMETRY(3001,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,0,6001,10,1,1),
  SDO_ORDINATE_ARRAY(0,1,0,0,65536,11,1852731203,1769235301,7632227,
                     1919397.1,680897.264,0))
```

The geometry's GTYPE is that of a standard three-dimensional point: 3001. Both the SRID and SDO_POINT fields are NULL. In the ELEM_INFO_ARRAY, the triplet 1,0,6001 describes a user-defined ETYPE (0) with an INTERPRETATION (6001), which GeoMedia interprets as its text format. The triplet 10,1,1 describes the location of the text as a point feature at OFFSET 10 with an ETYPE of 1 and an INTERPRETATION of 1.

```
SDO_GEOMETRY(3001,NULL,NULL,SDO_ELEM_INFO_ARRAY(1,0,6001, 10,1,1),
```

In the ORDINATE_ARRAY, the first value contains the rotation of the text about the rotation matrix; the next three values (0,0,1) represent the text rotation matrix. The fifth value (in this case, 65536) is an integer representation of the format and alignment of the text. The sixth value, in this case 11, is the number of characters of text being stored (including spaces). This is followed by integer representations of the actual text where each integer decomposes into 4 bytes of text. The last 3 values (1919397.1, 680897.264, 0) is the XYZ location of the text (based on the text justification point). Here is the breakdown:

```
SDO_GTYPE..................... 3001 - 3D POINT
SDO_SRID...................... NULL - No SRID Defined (projected data?)
SDO_POINT: NULL .............. SDO_POINT is NULL: Oriented point format used.
.....................................................................
SDO_ELEM_INFO_ARRAY
-> 1: 1....................... OFFSET: Index location in ordinate array
-> 2: 0....................... SDO_ETYPE: User defined element type
-> 3: 6001.................... SDO_INTERPRETATION: GeoMedia Text
-> 4: 10...................... OFFSET: Index location in ordinate array
-> 5: 1....................... SDO_ETYPE: Point
-> 6: 1....................... SDO_INTERPRETATION: # of points
.....................................................................
SDO_ORDINATES_ARRAY
-> 1: 0....................... Text Rotation (0 degrees ).
-> 2: 1....................... I Unit Vector
-> 3: 0....................... J Unit Vector
-> 4: 0....................... K Unit Vector
-> 5: 65536.................... Format: Rich Text (RTF) aligned Center Left.
-> 6: 11....................... Number of bytes of text
-> 7: 1852731203.............. Integer2Text -> Conn
-> 8: 1769235301.............. Integer2Text -> ecti
```

```
-> 9: 7632227.................. Integer2Text -> cut
-> 10: 1919397.1.............. X Ordinate
-> 11: 680897.264............. Y Ordinate
-> 12: 0...................... Z Ordinate
```

A text feature is treated just like a point feature, so entries are required in USER_SDO_GEOM_METADATA and in the GDOSYS metadata. In addition to the metadata, a spatial index is also required.

Raster Images

GeoMedia Professional has its own format for the storage of raster images and it also supports Oracle Spatial's GeoRaster format. Oracle's GeoRaster format, SDO_GEORASTER, was introduced in 10g. It is a component of Oracle spatial; you cannot use it with Oracle Locator.

When using its own format, GeoMedia Professional stores rasters similar to the way it stores text. GeoMedia stores the footprint or MBR of raster images as polygon features and includes some additional information about the display matrix and the full path to the image filename. The actual raster image is not stored in the database and is retrieved through its filename. An example is as follows:

```
SDO_GEOMETRY(3003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 0, 6002, 34, 1003, 1),
SDO_ORDINATE_ARRAY(1000, 0, 0, -2368572.33334194, 0, 1000, 0, 1558211.33334012,
0, 0, 1, 0, 0, 0, 0, 1, 31, 3801155, 5701724, 7471201, 6815845, 7667823,
6619251, 6029427, 5439573, 6357075, 7340141, 6619244, 7143497, 6750305, 3014757,
6881396, 102,
2262427.66665806, 1558211.33334012, 0, -2368572.33334194, 1558211.33334012, 0,
-2368572.33334194, -1293788.66665988, 0, 2262427.66665806, -1293788.66665988, 0,
2262427.66665806, 1558211.33334012, 0))
```

In this example, the ELEM_INFO_ARRAY triplet (1,0,6002) contains a user-defined ETYPE of 0 with an INTERPRETATION of 6002. This tells GeoMedia that the values it is reading correspond to a raster image. The second triplet (34,1003,1) represents the polygonal outline of the raster image. Here is the breakdown:

```
SDO_GTYPE..................... 3003 - 3D POLYGON
SDO_SRID...................... NULL - No SRID Defined (projected data?)
SDO_POINT: NULL .............. SDO_POINT is NULL: Oriented point format used.
.....................................................................
SDO_ELEM_INFO_ARRAY
-> 1: 1....................... OFFSET: Index location in ordinate array
-> 2: 0....................... SDO_ETYPE: User defined element type
-> 3: 6002.................... SDO_INTERPRETATION: GeoMedia Raster
-> 4: 34...................... OFFSET: Index location in ordinate array
-> 5: 1003.................... SDO_ETYPE: Outer Polygon
-> 6: 1....................... SDO_INTERPRETATION: Straight line segments
.....................................................................
SDO_ORDINATES_ARRAY
-> 1: 1000.................... Scale X
```

```
-> 2: 0..................... Matrix Padding
-> 3: 0..................... Matrix Padding
-> 4: -2368572.33334194..... Delta X
-> 5: 0..................... Matrix Padding
-> 6: 1000.................. Scale Y
-> 7: 0..................... Matrix Padding
-> 8: 1558211.33334012..... Delta Y
-> 9: 0..................... Matrix Padding
-> 10: 0..................... Matrix Padding
-> 11: 1..................... Scale Y
-> 12: 0..................... Delta Z
-> 13: 0..................... Matrix Padding
-> 14: 0..................... Matrix Padding
-> 15: 0..................... Matrix Padding
-> 16: 1..................... Matrix Padding
Moniker/location in Rich Text/Unicode format:
-> 17: 31.................... Length of Moniker
-> 18: 3801155................ Integer2Text -> C :
-> 19: 5701724................ Integer2Text -> \ W
-> 20: 7471201................ Integer2Text -> a r
-> 21: 6815845................ Integer2Text -> e h
-> 22: 7667823................ Integer2Text -> o u
-> 23: 6619251................ Integer2Text -> s e
-> 24: 6029427................ Integer2Text -> s \
-> 25: 5439573................ Integer2Text -> U S
-> 26: 6357075................ Integer2Text -> S a
-> 27: 7340141................ Integer2Text -> m p
-> 28: 6619244................ Integer2Text -> l e
-> 29: 7143497................ Integer2Text -> I m
-> 30: 6750305................ Integer2Text -> a g
-> 31: 3014757................ Integer2Text -> e .
-> 32: 6881396................ Integer2Text -> t i
-> 33: 102.................... Integer2Text -> f.
Raster Footprint – Standard 4 point polygon:
-> 34: 2262427.66665806.......
-> 35: 1558211.33334012.......
-> 36: 0......................
-> 37: -2368572.33334194......
-> 38: 1558211.33334012.......
-> 39: 0......................
-> 40: -2368572.33334194......
-> 41: -1293788.66665988......
-> 42: 0......................
-> 43: 2262427.66665806.......
-> 44: -1293788.66665988......
-> 45: 0......................
-> 46: 2262427.66665806.......
-> 47: 1558211.33334012.......
-> 48: 0......................
```

The first 16 values in the ORDINATE_ARRAY store the display matrix for the raster image. The 17th value stores the length of the filename/moniker in bytes. The integers following the length are the actual filenames decomposed into four-byte integers. The number of integers used will depend on the length of the filename. This is followed by the five coordinate pairs that make up the polygonal outline of the raster image (the first and last coordinate are the same).

By treating the raster image outline as a polygon feature, GeoMedia can make full use of Oracle's spatial filtering capability. Like any other geometry, entries are required in Oracle's USER_SDO_GEOM_METADATA and a spatial index should be present. Entries in GDOSYS metadata should show this as a GraphicCoverage data type in GeoMedia. in the GDOSYS metadata

GeoRaster is a feature of Oracle Spatial (not locator) that will let you store, index, query, and analyze any raster image or gridded data. Oracle uses two new database objects to store raster images: SDO_GEORASTER and SDO_RASTER. The use of GeoRaster data in GeoMedia should be transparent to the user as long as the standard raster metadata is assigned to the table containing the SDO_GEORASTER datatype. GeoMedia Professional treats GeoRaster data as read-only. GeoMedia Professional will not write to Oracle's GeoRaster format but will load and display the data if it is available.

Data Type Matching - Oracle to GeoMedia

During a connection, the column data types used in Oracle are mapped to internal GeoMedia data objects (known as GDO) by the Oracle Object Model data server. The following table identifies the mapping used when making the conversion. **Any data type that is not listed in this table is considered unsupported** and will be ignored by GeoMedia applications.

Oracle Data Type	GeoMedia Data Type
CHAR NCAR VARCHAR2 NVARCHAR2	Text (<= 255) Memo (> 255)
DATE	Date
NUMBER	Double
NUMBER(p,s)	Long (p<10>1, s= 0) Double (s != 0) Double (p >= 10) Double (p=0, s=0) Boolean (p=1,s=0)
INTEGER (p is null, s = 0)	Long
FLOAT	Double
BLOB	LongBinary

Oracle Data Type	GeoMedia Data Type
CLOB NCLOB	Memo
SDO_GEOMETRY	Spatial
SDO_GEORASTER	Coverage

These default mappings may be overridden by GeoMedia client metadata tables. Unsupported column data types will be ignored by GeoMedia. There has been concern expressed over the fact that GeoMedia converts NUMBER(10) in Oracle to a DOUBLE in GDO rather than a LONG. This conversion is required because NUMBER(10) can store a value that will not fit into the LONG datatype:

- Oracle NUMBER(10): +-9999999999
- GeoMedia LONG: +-2147483648

Using DOUBLE in this case prevents possible data loss.

Any INTEGER based column that is mapped to Autonumber in GeoMedia will require a sequence to be assigned. You can assign sequences to other NUMBER types, but they will not be treated as autonumber.

Data Type Matching – GeoMedia to Oracle

The following table identifies the mapping used when converting from GeoMedia's GDO data types to Oracle data types (this applies only to metadata changes through the read-write data server).

GeoMedia Data Type	Oracle Data Type
Boolean	NUMBER (1,0)
Byte	NUMBER (3,0)
Integer	INTEGER
Long	INTEGER
Single	FLOAT
Double	FLOAT
Currency	FLOAT
Date	DATE
Text	VARCHAR2
LongBinary	BLOB
Memo	CLOB
Guid	VARCHAR2

GeoMedia Data Type	Oracle Data Type
Spatial/Graphic (any type)	MDSYS.SDO_GEOMETRY

Native Geometry Metadata

Each column of type SDO_GEOMETRY requires range and tolerance information for each dimension to be specified in a global table owned by Oracle's MDSYS schema. All read-write operations related to Oracle's spatial metadata should utilize the USER_SDO_GEOM_METADATA view and be run by the owner of the tables/views involved. To retrieve the metadata information based on the user's privileges, use the read-only view ALL_SDO_GEOM_METADATA.

Both Oracle and GeoMedia require that all SDO_GEOMETRY columns in a given schema have appropriate values in the USER_SDO_GEOM_METADATA view. GeoMedia takes this one step further and requires views to also have entries here as well. Behind the scenes, GeoMedia writes metadata information to USER_SDO_GEOM_METADATA but reads the metadata through ALL_SDO_GEOM_METADATA. For this reason, GeoMedia can only create tables in the schema owned by the connected user, but it can read metadata from any schema the connected user has privileges to access.

Note: DBA's can work around this by granting insert privileges on MDSYS.SDO_GEOM_METADATA_TABLE to the affected user. This will allow GeoMedia to create tables in schemas other than the connected user.

If you use the **Output to Feature Classes** command, USER_SDO_GEOM_METADATA is populated automatically using default values stored in the GDOSYS.GPARAMETERS table. If you use the **Export to Oracle Object Model** command, default values are provided in the <FEATURE NAME>_POST.SQL files. In all other cases, you will need to populate this view manually.

You can set default values yourself using the following insert statements in SQL:

```
INSERT INTO USER_SDO_GEOM_METADATA VALUES
  ('TABLE_NAME','GEOMETRY_COLUMN',
    MDSYS.SDO_DIM_ARRAY (
         MDSYS.SDO_DIM_ELEMENT('X',-2147483648,2147483647,0.00005),
         MDSYS.SDO_DIM_ELEMENT('Y',-2147483648,2147483647,0.00005)),
    NULL);
```

If your data is three dimensional, you need to include a dimension statement for the Z-value as follows:

```
INSERT INTO USER_SDO_GEOM_METADATA VALUES
  ('TABLE_NAME','GEOMETRY_COLUMN',
    MDSYS.SDO_DIM_ARRAY (
         MDSYS.SDO_DIM_ELEMENT('X',-2147483648,2147483647,0.00005),
         MDSYS.SDO_DIM_ELEMENT('Y',-2147483648,2147483647,0.00005),
         MDSYS.SDO_DIM_ELEMENT('Z',-2147483648,2147483647,0.00005)),
    NULL);
```

The default values assigned by GeoMedia should fit data for any projection and the tolerance matches what GeoMedia uses internally.

For geodetic data using longitude and latitude coordinates, the metadata range MUST be +/- 180 and +/-90 similar to that shown below:

```
INSERT INTO USER_SDO_GEOM_METADATA VALUES
  ('TABLE_NAME','GEOMETRY_COLUMN',
    MDSYS.SDO_DIM_ARRAY(MDSYS.SDO_DIM_ELEMENT('X',-180,180,0.05),
                        MDSYS.SDO_DIM_ELEMENT('Y',-90,90,0.05)),
    NULL);
```

Failure to assign the correct metadata range for geodetic data will lead to incorrect results for both spatial filter queries and spatial indexing.

The tolerance value used in the SDO_DIM_ARRAY is used by Oracle to determine when distinct coordinate values are to be considered equal and to associate a level of precision with spatial data. This value must be a positive number greater than zero.

For non-geodetic data, the tolerance value is in the unit of storage associated with the data. For example, if the unit of measurement is meters, a tolerance value of 0.0001 indicates a tolerance of 1/10 millimeter. This is equivalent to GeoMedia's built-in spatial tolerance. For best results, Oracle's tolerance should be no less than half of GeoMedia's, or 0.00005 meters. Keep this in mind if your data is stored in feet because conversion will apply.

For geodetic data (longitude, latitude), the tolerance value is always in meters (a tolerance of 1 would mean 1 meter). The tolerance value for geodetic data should never be smaller than 0.05 (5 centimeters), and in most cases, it should be larger. If a tolerance value less than 0.05 is specified, Oracle will automatically default it to 0.05.

Oracle uses the tolerance in its own internal calculations to determine the relationship between vertices; it has no effect on GeoMedia's own tolerance or on client side calculations. However, tolerance is important when using spatial analysis filters such as INSIDE, EQUAL, or COVEREDBY.

Oracle Coordinate Systems - SRID

Oracle supports a variety of coordinate systems that can be assigned to spatial data. These coordinate systems can be listed by means of the following query:

```
SELECT CS_NAME, SRID FROM MDSYS.CS_SRS ;
```

To use one of Oracle's coordinate systems, the SRID of the coordinate system must be loaded into both the USER_SDO_GEOM_METADATA view and the SDO_SRID component of the geometry column containing the spatial data. For example, given a feature class called ROADS containing a GEOMETRY column that requires a SRID of 8307, you could use the following to populate the required column:

```
UPDATE USER_SDO_GEOM_METADATA SET SRID=8307
 WHERE TABLE_NAME='ROADS'
   AND COLUMN_NAME='GEOMETRY';
```

```
UPDATE ROADS A SET A.GEOMETRY.SDO_SRID=8307;
```

For the SRID to function correctly, it has to be present in both locations. Once the SRID is set, the spatial indexes will need to be re-created on the affected geometry column.

GeoMedia Professional ignores the SRID value used by Oracle because it already has built-in support for most coordinate systems. There is no direct mapping between Oracle's SRID's and GeoMedia's coordinate systems. You can use the SRID for compatibility with other applications, and GeoMedia automatically will pass through the value like any other attribute.

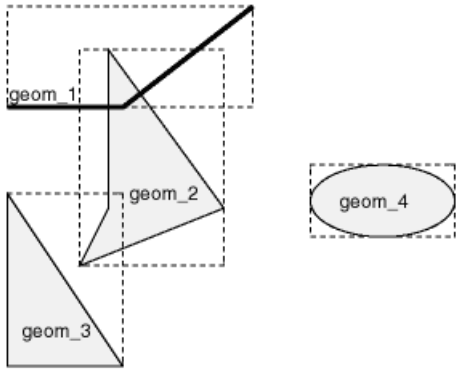
In most cases, the SRID is not required and can be left as NULL. In the case of geodetic or geographic data, Oracle requires the use of the SRID for its own internal calculations with spatial filters and indexes. If you are using geographic data, you must remember to set the SRID to the correct value. For example, if your data uses geographic coordinates with a datum of WGS84, you could use the following query to bring up a list of available SRIDs:

```
COLUMN CS_NAME FORMAT A60;
SELECT CS_NAME ,SRID
  FROM MDSYS.CS_SRS
 WHERE CS_NAME LIKE 'LONG%'
   AND (CS_NAME LIKE '%NAD%' OR CS_NAME LIKE '%WGS%');
```

The SRID to use in this example is 8307.

SPATIAL INDEXING AND ANALYSIS

Both GeoMedia and Oracle require spatial indexes to be present on all SDO_GEOMETRY-based columns present in the database. Oracle uses domain based R-Tree indexing which approximates stored geometry by using a minimum bounding rectangle, or MBR, for each geometry.



For any given feature class (or layer), the R-Tree index consists of a hierarchical index on all the MBRs of the geometries in the feature class.

Creating Spatial Indexes

Creating a spatial index is very similar in syntax to creating an index on a column of a simple data type. For tables created using GeoMedia's Feature Class Definition, the spatial index will be created for you. If you create your own tables, you need to create the spatial index manually using the example syntax shown below:

```
CREATE INDEX <index_name>
          ON <tablename> (<sdo_geometry_col>)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

This is the syntax in its simplest form; there are many other parameters available depending on how you want the index to behave, including the ability to choose the index tablespace or to optimize the index creation. These are covered thoroughly in Oracle's documentation.

FOR MORE DETAILED, UP-TO-DATE INFORMATION, SEE THE ORACLE SPATIAL USERS GUIDE and REFERENCE.

Spatial indexes are made up of several different database objects. If the CREATE INDEX statement fails for any reason, you may end up with a partial spatial index. Attempts to re-index or to delete the existing index will return an error stating that the index is marked as LOADING/ UNLOADING. If this occurs, you will need to add the FORCE keyword to the DROP INDEX command. Here is an example:

```
DROP INDEX <index_name> FORCE;
```

Oracle allows geometries to contain up to four dimensions. The GeoMedia Object Model Data Server supports both 2-D and 3-D data for read-write operations, but only read-only operations on 4-D data (LRS). All data is served to the client as 3-D. In the case of 2-D, the third dimension is assigned a value

of zero when reading the data. In the case of 4-D Oracle data, the fourth dimension is ignored when reading.

Spatial indexes can operate in all three dimensions, but GeoMedia will only work with 2-D spatial indexes.

Spatial Filtering

Spatial filtering is critical when using large datasets. The more limitations placed on the amount of data passed to the client, the better the performance will be. When using the Oracle, all spatial filtering initiated within GeoMedia is actually processed on the database server. This includes both the coarse first-pass filter and the finer second-pass filter. Attempting to use a spatial filter without having the required spatial indexes will result in the following error:

```
Recordset is invalid.
MORE:
ORA-13226: Interface not supported without a spatial index.
```

The spatial filter operators in GeoMedia Professional are mapped to the Oracle spatial filters in the following way (listed in order of performance):

GeoMedia Filter Type	Oracle Filter Type
Coarse Overlap	MDSYS.SDO_FILTER
Overlap	MDSYS.SDO_RELATE Mask: ANYINTERACT
Entirely Inside	MDSYS.SDO_RELATE Mask: INSIDE
Inside	MDSYS.SDO_RELATE Mask: INSIDE+COVEREDBY+EQUAL

Coarse Overlap is the fastest performing filter because it uses Oracle's single pass SDO_FILTER function, but the results may exceed the boundary of the filter area. This is the best filter to use when performance is the only consideration. The other three filters are treated as standard spatial queries with the filter area being passed to Oracle's SDO_RELATE operator in a bind variable. SDO_RELATE uses a two-pass filter and produces more accurate results. Of the SDO_RELATE based filters, INSIDE is the slowest because it is processing a union of three different filter operations. INSIDE will also give the best visual results because it will return everything up to and including the boundary of the filter area.

Native Queries

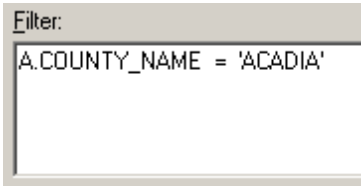
GeoMedia's **Native query** command builds and passes native spatial queries directly to Oracle for processing using Oracle SDO_RELATE function. This is very similar to the spatial filter command except that, in this case, two feature classes are used as input. Depending on the amount and type of data being processed, this may be faster than using GeoMedia's client side spatial queries.

There are three main differences between the spatial querying capability in GeoMedia and that used by Oracle:

- **Topology Engine** – GeoMedia and Oracle use completely different topology algorithms.
- **Client versus Server** – Spatial queries in GeoMedia operate completely on the client, while native queries run on the server. This is an advantage if the data is quite large and you want to leverage the server's power for the processing stage.
- **Distinct Results** – GeoMedia's spatial queries use a DISTINCT operator that ensures that results are not redundant. For example, you have a warehouse consisting of a single County feature class that entirely contains three River feature classes. When GeoMedia performs a spatial query on all the counties that entirely contain rivers, the process returns one county. The same native spatial query would return three counties, one for each of the three rivers, even though it is the same county being returned each time. Both answers are correct. GeoMedia just returns what is required to display the result while Oracle returns the numeric result.

If you are using attribute filters with your native queries, you must manually insert a table alias identifier in the query statement. Filters on the **Select Features in table** must be prefixed with **A**, and **B** must prefix filters on the second **Features in table**. The **Filter** dialog box will let you interactively enter columns, operators, and values, but it is up to you to enter the appropriate table alias whether it is **A** or **B**.

In the following example, an attribute filter is being applied to the **Select Features in** table. The prefix A. has to be added to the query string in order for the attribute query to be properly applied. The result is A. COUNTY_NAME:



GEOMEDIA'S GDOSYS METADATA SCHEMA

GeoMedia uses a metadata schema called GDOSYS. For read-only connections, a GDOSYS schema is not required, but it is recommended (for coordinate systems). For read-write connections, the GDOSYS user must exist in the Oracle database instance and must contain the set of required metadata tables.

GeoMedia's Object Model Data Server uses GDOSYS.GALIASTABLE to determine if the GDOSYS schema is to be used. If GALIASTABLE is found during the initial connection, all tables and views in the schema will require appropriate entries in the GDOSYS metadata tables even if the schema is read-only.

Creating the GDOSYS Schema

The GDOSYS metadata schema is created in your Oracle instance by a SQL script that is accessed via GeoMedia Professional's **Database Utilities**. To create GDOSYS, you need to connect to your database as the DBA using **Database Utilities** and run the **Create Metadata Tables** command. This only needs to be done once per Oracle instance. Make sure you use a net service name when connecting even if the database instance is local. If you do not use the net service name, GDOSYS will not be created and no warning message will be issued. For best results, you should put GDOSYS in its own tablespace. Because this is primarily attribute data, a 4k block size is sufficient but the default of 8k is adequate as well.

You can also use **Database Utilities** to update GDOSYS to the current version. Connect as a DBA user and select the **Create Metadata Tables** button. Because GDOSYS already exists, the process verifies the schema and updates it as necessary.

Oracle DBA's can use the CREATEGDOSYS.SQL script in the ..GEOMEDIA PROFESSIONAL/SCRIPTS folder to manually create the GDOSYS schema. The script will need to be run from SQL*Plus while connected as a system DBA. The syntax of the create metadata script is:

```
@CreateGDOSYS <NET_SERVICE> <GDOSYS_PSWD> <ORA_PROFILE>
              <USER_TABLESPACE> <TEMP_TABLESPACE>
```

where:

- <NET_SERVICE> is the Oracle service name.
- <GDOSYS_PSWD> is the password to use for the GDOSYS user.
- <ORA_PROFILE> is the profile to use when creating the user account.
- <USER_TABLESPACE> is the tablespace where the schema will be created.
- <TEMP_TABLESPACE> is the temporary tablespace to use.

For example:

```
@CreateGDOSYS ORCL GDOSYS DEFAULT USERS TEMP
```

You can also update GDOSYS to the current version using the UPDATEGDOSYS.SQL script in the ../GEOMEDIA PROFESSIONAL/SCRIPTS folder. You must connect as DBA to run this script as well.

The current GDOSYS is backward compatible to GeoMedia Professional 5.2; however, you must update GDOSYS with every new release of GeoMedia Professional.

The Default GDOSYS Schema Definition

Whether you use the scripts or **Database Utilities** to create the GDOSYS metadata schema, the following objects are created:

GDOSYS Object	Type
ATTRIBUTEPROPERTIES	Table
FIELDLOOKUP	Table
GALIASTABLE	Table
GCOORDSYSTEM	Table
GEOMETRYPROPERTIES	Table
GEXCLUSIONS	Table
GFEATURES	View
GFEATURESBASE	Table
GFIELDMAPPING	Table
GINDEXCOLUMNS	Table
GPARAMETERS	Table
GPICKLISTS	Table
GQUEUE	View
GQUEUEBASE	Table
LIBRARYTABLES	Base
LIBRARYTABLESBASE	Table
MODIFIEDTABLES	Table
MODIFICATIONLOG	Table
GMODLOG	Sequence

GDOSYS Object	Type
GAUTONUMBERSEQUENCE	Sequence
FIELDLOOKUPINDEXID1	Sequence
DELETEMETADATAGMT	Trigger

ATTRIBUTEPROPERTIES

The ATTRIBUTEPROPERTIES metadata table describes the attribute types for the columns listed in the FIELDLOOKUP table. The common link between this table and FIELDLOOKUP is the INDEXID primary key column. The definition of the table is:

Name	Virtual	Type	Nullable
ISKEYFIELD	☐	INTEGER	☑
FIELDDESCRIPTION	☐	VARCHAR2(255)	☑
INDEXID	☐	INTEGER	☐
FIELDFORMAT	☐	VARCHAR2(255)	☑
FIELDTYPE	☐	INTEGER	☑
ISFIELDDISPLAYABLE	☐	INTEGER	☑
FIELDPRECISION	☐	INTEGER	☑

- ISKEYFIELD – Tells GeoMedia applications whether the column indicated by the INDEXID is a primary key. The default value is 0 for FALSE, any other value (generally -1) is TRUE.
- FIELDDESCRIPTION – A user-provided description of the column.
- FIELDTYPE – Determines how GeoMedia applications interpret the data type used in the column definition (based on the data type matching table listed earlier). Available field type values are:

```
1 – Boolean          8 – Date
2 – Byte             10 - Text
3 – Integer          11 - Binary
4 - Long             12 - Memo
5 - Currency         15 - GUID
6 - Single           32 - Spatial geometry
7 - Double           33 – Graphic geometry
```

- FIELDFORMAT – Determines the general format of the data being displayed. Format types depend on the FIELDTYPE and can include GENERAL NUMBER, DATE/TIME, YES/NO, TRUE/FALSE, ETC., ETC.
- FIELDPRECISION – Represents the number of decimal places exposed in GeoMedia Professional. For numeric data types, the default is 6.
- ISFIELDDISPLAYABLE – Determines whether a column is displayed in GeoMedia Professional. The default value is -1 (TRUE) so the column is displayed, use 0 (zero) to hide the column in GeoMedia.

FIELDLOOKUP

The FIELDLOOKUP metadata table provides a unique identifier (INDEXID) for every column in every table that can be utilized as a feature class by GeoMedia applications. These identifiers are then used by other metadata tables. The definition of the table is:

Name	Virtual	Type	Nullable
INDEXID	☐	INTEGER	☐
FEATURENAME	☐	VARCHAR2(255)	☐
FIELDNAME	☐	VARCHAR2(255)	☐

- INDEXID – This primary key column stores a unique integer identifier for every column in every table/view used by GeoMedia applications. This allows other metadata tables such as ATTRIBUTEPROPERTIES and GEOMETRYPROPERTIES to reference feature class names from a common source. The INDEXID value is generated from the GDOSYS sequence FIELDLOOKUPINDEXID1.
- FEATURENAME – Contains the table/view name in the format OWNER.TABLE. This format is required.
- FIELDNAME – Contains each of the column names that are in the associated field name.

GALIASTABLE

The GALIASTABLE metadata table determines the names used by other metadata tables. It must be located in the GDOSYS schema, and it must have the specific name GALIASTABLE. All other metadata tables are referenced through GALIASTABLE. During a database connection, if a given user has select privileges on GDOSYS.GALIATABLE, that user will require metadata GeoMedia feature classes regardless of connection type (read-write/read-only). The definition of this table is:

Name	Virtual	Type	Nullable
TABLETYPE	☐	VARCHAR2(255)	☐
TABLENAME	☐	VARCHAR2(255)	☑

- TABLETYPE – GeoMedia's required name.
- TABLENAME – Actual metadata table name as stored. The table names in this field must be in the format OWNER.TABLE. For example, GDOSYS.GCOORDSYSTEM.

GCOORDSYSTEM

GCOORDSYSTEM stores coordinate system definitions. If this table is not present, no coordinate system transformation will occur, and the GeoWorkspace coordinate system will be used. This table is not typically user editable because of the large number of columns and types of parameters required to define a coordinate system. There are four columns worth noting:

Name	Virtual	Type	Nullable
CSGUID	☐	VARCHAR2(38)	☐
CSGUIDTYPE	☐	INTEGER	☑
NAME	☐	VARCHAR2(100)	☑
DESCRIPTION	☐	VARCHAR2(255)	☑

- CSGUID – This primary key column uses a special UID to uniquely identify the coordinate system parameters defined in each row. The value is also used by GEOMETRYPROPERTIES and in GFIELDMAPPING to associate the coordinate system parameters with a GeoMedia feature class.
- CSGUIDTYPE – An indicator for whether the coordinate system is permanent or not. This is really used by the GeoMedia API. In the database table, it should always be set to 2.
- NAME – The name the user has assigned to this coordinate system. It is an optional parameter but should be used because it makes the coordinate system easier to identify, particularly in the Oracle environment.
- DESCRIPTION – An optional user-provided description of the coordinate system.
- The other attribute columns in GCOORDSYSTEM are not user editable and should not be modified in any way.

Coordinate systems should be created through the GeoMedia Professional **Define Coordinate System** command. When a defined coordinates system is assigned to a feature class, the parameters that make up the coordinate system are inserted into the database table. Any feature class that uses the coordinate system is assigned the CSGUID for that coordinate system.

Coordinate systems are defined on a per-feature-class basis. Each feature class can have its own coordinate system. The easiest way to assign a coordinate system to a feature class is by using **Database Utilities**, which are available in the GeoMedia program group. If you have incorrectly assigned a coordinate system to a feature class, you can also use the **Database Utilities** to correct the assigned coordinate system.

SEE THE SECTION ON "DATABASE UTILITIES."

GDOSYS.GCOORDSYSTEM stores all of the coordinate systems used by all feature classes in all schemas in the Oracle database instance. This allows any user with the proper privileges to access and to display geometry data from any schema. Because all coordinate systems are available, you may find it useful to designate a default coordinate system for each schema to use. Default coordinate systems can be assigned using **Database Utilities** or the **Feature Class Definition** command. Only one default coordinate system is allowed per schema. The CSGUID of the default coordinate system is stored in the GDOSYS.GPARAMETERS table along with the schema name. It is a best practice to always set a default coordinate system.

When digitizing in GeoMedia Professional, you should ensure that the GeoWorkspace coordinate system matches the coordinate system of the feature class into which you are digitizing. There are exceptions but it is considered a best practice to avoid coordinate system transformations during data entry. GeoMedia Professional will compare the GeoWorkspace coordinate system to the coordinate system of the feature selected for editing and will warn the user if there is a mismatch. It will be up to the user to determine whether they want to proceed or rectify the mismatch.

GEOMETRYPROPERTIES

The GEOMETRYROPERTIES metadata table stores the geometry type, primary geometry flag, and the coordinate system ID for the SDO_GEOMETRY fields in each feature classes. The common link between

this table and FIELDLOOKUP is the INDEXID column. This table determines the coordinate system that is assigned to each feature class. The definition of this table is:

Name	Virtual	Type	Nullable
PRIMARYGEOMETRYFLAG	☐	INTEGER	☑
GEOMETRYTYPE	☐	INTEGER	☑
GCOORDSYSTEMGUID	☐	VARCHAR2(255)	☑
FIELDDESCRIPTION	☐	VARCHAR2(255)	☑
INDEXID	☐	INTEGER	☐

- PRIMARYGEOMETRYFLAG – A feature class can contain multiple geometry columns, but only one column is allowed to be editable from inside GeoMedia, this is the primary geometry. A value of -1 (True) means the geometry column is the primary geometry column. All other geometry columns in the feature class should be assigned 0 (False). Only one primary geometry field is allowed.
- GEOMETRYTYPE – This field determines how the data server maps the geometry:

```
1 – Line                2 – Area
3 – AnySpatial          4 – Coverage
5 – GraphicsText       10 – Point
```

- Refer to the section on Oracle to GeoMedia Element Type conversion for more information on the values used here.
- GCOORDSYSTEMGUID – Contains the CSGUID from the GCOORDSYSTEM table. It tells the data server what coordinate system is assigned to the geometry. Each geometry column can be assigned a different coordinate system, if required, as long as the indicated CSGUID is present in the GCOORDSYSTEM
- FIELDDESCRIPTION – Contains an optional user-provided description of the geometry.
- INDEXID – This primary key column links the information to the actual feature class/geometry column defined in the FIELDLOOKUP table.

GEXCLUSIONS

The GEXCLUSIONS metadata table is specific to the Oracle Object Model Data Server and is used to exclude schemas from the initial connection scan. When establishing an Oracle connection, any schema that the connected user has privileges to see will be scanned for compatibility. This includes any schema granted to PUBLIC and all public dictionary tables and views. The more schemas that are available to the connected user, the longer the connection takes. This is one reason it is not recommended to connect as a user with the DBA role.

The schemas listed in GEXCLUSIONS are not hidden from GeoMedia; they are only used to exclude schemas from the initial connection scan. If the GEXCLUSIONS table does not exist, the data server will assume a hard-coded list of Oracle schemas to exclude. If the table does exist, only those schemas listed in the table will be excluded. The default list of schemas to exclude if the table is not present is as follows: CTXSYS, MDSYS, OLAPSYS, ORDSYS, OUTLN, SH, SYS, SYSTEM, WMSYS, WKPROXY, WKSYS, and XDB. The definition of this table is:

Name	Virtual	Type	Nullable
OWNER	☐	VARCHAR2(30)	☐
EXTYPE	☐	VARCHAR2(1)	☐

- OWNER – Schema name to exclude.
- EXTYPE – Specifies what type of exclusion to make. Setting EXTYPE to 'A' will exclude all views and tables from this schema. Setting the EXTYPE to 'V' will exclude only views, but will include tables.

GFEATURES and GFEATURESBASE

The GFEATURESBASE metadata table stores the names of all the tables and views available to GeoMedia applications as feature classes. GeoMedia access this table via the GFEATURES view which lists all feature classes available to the connected user (anything the user has select privileges on). GFEATURES is called frequently and is used to populate GeoMedia dialogs that list feature classes. GFEATURESBASE is defined as:

Name	Virtual	Type	Nullable
FEATURENAME	☐	VARCHAR2(255)	☐
GEOMETRYTYPE	☐	INTEGER	☑
PRIMARYGEOMETRYFIELDNAME	☐	VARCHAR2(255)	☑
FEATUREDESCRIPTION	☐	VARCHAR2(255)	☑

- FEATURENAME – This primary key column is the name of the feature class in the format OWNER.FEATURENAME. This format is required.
- GEOMETRYTYPE – Determines how the data server maps the geometry:

```
1 – Line            2 – Area
3 – AnySpatial      4 – Coverage
5 – GraphicsText   10 – Point
```

Refer to the section on Oracle to GeoMedia Element Type conversion for more information on the values used here.

- PRIMARYGEOMETRYFIELDNAME – Contains the name of the primary geometry column.
- FEATUREDESCRIPTION – Optionally contains a user-provided description of the FEATURENAME.

The standard definition of the GFEATURES view is as follows:

```
SELECT * FROM GDOSYS.GFEATURESBASE
 WHERE EXISTS
 (SELECT 1
    FROM ALL_OBJECTS
   WHERE OWNER=SUBSTR(FEATURENAME,1,INSTR(FEATURENAME,'.',1)-1)
     AND OBJECT_TYPE IN ('TABLE','VIEW')
     AND OBJECT_NAME=SUBSTR(FEATURENAME,INSTR(FEATURENAME,'.',1)+1));
```

This view only serves up the feature classes that the connected user has privileges to select. All other feature classes are hidden. You can set up the GFEATURES view as you see fit; however, it must have the same definition as the GFEATURESBASE table. If you are using only one GeoMedia warehouse or

want to expose all GeoMedia feature classes to all users, you do not need to use the view; simply rename GFEATURESBASE to GFEATURES.

In Oracle 12c, you may see a performance improvement by changing the definition of GFEATURES to the following:

```
CREATE OR REPLACE VIEW GDOSYS.GFEATURES AS
SELECT FEATURENAME,GEOMETRYTYPE,
       PRIMARYGEOMETRYFIELDNAME,FEATUREDESCRIPTION
  FROM GDOSYS.GFEATURESBASE
 WHERE EXISTS
(SELECT 1
   FROM ALL_OBJECTS
  WHERE ORACLE_MAINTAINED='N'
    AND OBJECT_TYPE IN ('TABLE','VIEW')
    AND OWNER = SUBSTR(FEATURENAME,1,INSTR(FEATURENAME,'.',1)-1)
    AND OBJECT_NAME = SUBSTR(FEATURENAME,INSTR(FEATURENAME,'.',1)+1));
```

GFIELDMAPPING

The GFIELDMAPPING metadata table is specific to the Oracle Object Model Data Server, and it is not used directly by GeoMedia. The values in this table are used to override various aspects of column data type definitions. Information stored here typically consists of the primary key column and the primary geometry with their associated GeoMedia data types, coordinate system IDs, and any assigned sequences. The definition of this table is:

Name	Virtual	Type	Nullable
OWNER	☐	VARCHAR2(30)	☐
TABLE_NAME	☐	VARCHAR2(30)	☐
COLUMN_NAME	☐	VARCHAR2(30)	☐
DATA_TYPE	☐	INTEGER	☐
DATA_SUBTYPE	☐	INTEGER	☑
CSGUID	☐	VARCHAR2(38)	☑
SEQUENCE_OWNER	☐	VARCHAR2(30)	☑
SEQUENCE_NAME	☐	VARCHAR2(30)	☑

- OWNER, TABLE_NAME, AND COLUMN_NAME – Multi-column primary key:
- OWNER – Contains the owner of the table or feature class.
- TABLE_NAME – Contains the name of the table or feature class.
- COLUMN_NAME – Contains the column in the table that this information applies to.
- DATA_TYPE – Determines how GeoMedia interprets the datatype used in the column definition. Field type values include:

```
1 – Boolean        8 – Date
2 – Byte           10 - Text
3 – Integer        11 - Binary
4 - Long           12 - Memo
```

```
5 - Currency        15 - GUID
6 - Single          32 - Spatial geometry
7 - Double          33 – Graphic geometry
```

SEE THE "GEOMEDIA TYPE MAPPING" SECTION FOR MORE INFORMATION ON WHAT DATATYPES TO USE HERE.

- DATA_SUBTYPE – Used when the DATA_TYPE is 32 or 33; the subtype determines the graphic type.

```
1 – Linear          2 – Areal
3 – AnySpatial      4 – Coverage
5 - GraphicsText    10 - Point
```

SEE THE "GEOMEDIA TYPE MAPPING" SECTION FOR MORE INFORMATION ON THE VALUES USED HERE.

- CSGUID – Contains the coordinate system assigned to the primary geometry field. It corresponds to the CSGUID in the GCOORDSYSTEM table.
- SEQUENCE_OWNER – Contains the owner of the sequence used for the autonumber field.
- SEQUENCE_NAME – Contains the name of the sequence assigned to the autonumber field.

GINDEX_COLUMNS

The GINDEX_COLUMNS metadata table is specific to the Oracle Object Model Data Server and is used to specify the pseudo primary key (unique column) in a database view that is seen as a feature class in GeoMedia. GeoMedia requires that feature classes have a unique and not null column that can be used when retrieving data. Normally a primary key is used but database views do not have key columns. This table is populated using **Database Utilities**. The only limitation to using GINDEX_COLUMNS is that only one column may be specified as a primary key for the view. The definition of this table is:

Name	Virtual	Type	Nullable	Default/Expr.
OWNER	☐	VARCHAR2(30)	☐	
OBJECT_NAME	☐	VARCHAR2(30)	☐	
INDEX_NAME	☐	VARCHAR2(30)	☐	
INDEX_TYPE	☐	VARCHAR2(2)	☑	'P'
COLUMN_NAME	☐	VARCHAR2(30)	☐	
COLUMN_POSITION	☐	INTEGER	☑	1

- OWNER, OBJECT_NAME, INDEX_NAME, COLUMN_NAME make up the multi-column primary key:
- OWNER – Contains the owner of the view.
- OBJECT_NAME – Contains the name of the view.
- INDEX_NAME – Contains the primary or unique key index name from the base table. The key must be included in the view definition and be key preserved.
- COLUMN_NAME – Contains the name of the column in the view that will use the index in INDEX_NAME. The index name is just a place holder and does not reflect an actual index.
- INDEX_TYPE – Contains the type of the index, 'P' for primary, 'U' for Unique.

- COLUMN_POSITION – Contains the position of the key column as defined in the view.

If a view does not have a key defined in the GINDEX_COLUMNS, the Oracle Object data server will attempt to determine the underlying table for the view and will use any primary key defined for that table. If it fails to find any column to use, the view will be treated as a read-only snapshot which may have a negative impact on performance for this feature class.

GPARAMETERS

The GPARAMETERS metadata table contains parameter/value pairs. GPARAMETERS is used by the data server when a geometry based column is created in an Oracle schema. These values are used to set various parameters in Oracle as the geometry field is created and indexed. The table is defined as:

Name	Virtual	Type
GPARAMETER	☐	VARCHAR2(255)
GVALUE	☐	VARCHAR2(255)

GPARAMETERS comes prepopulated with the following values:

GParameter	Type	Default GValue	Comments
SpatialIndexLevel	Long	1	(no longer used)
NumberOfTiles	Long	1	(no longer used)
MaxLevel	Long	1	(no longer used)
X_LowerBound	Double	-2147483648	Xlo value for USER_SDO_GEOM_METADATA
X_UpperBound	Double	2147483647	Xhi value for USER_SDO_GEOM_METADATA
X_Tolerance	Double	.00005	X1=X2 if |X1-X2| < X_Tolerance
Y_LowerBound	Double	-2147483648	Ylo value for USER_SDO_GEOM_METADATA
Y_UpperBound	Double	2147483647	Yhi value for USER_SDO_GEOM_METADATA
Y_Tolerance	Double	.00005	Y1=Y2 if |Y1-Y2| < Y_Tolerance
Z_LowerBound	Double	-2147483648	Zlo value for USER_SDO_GEOM_METADATA
Z_UpperBound	Double	2147483647	Zhi value for USER_SDO_GEOM_METADATA
Z_Tolerance	Double	.00005	Z1=Z2 if |Z1-Z2| < Z_Tolerance

Although GVALUE values are strings, the types listed above are what the string is converted to during processing.

If this table does not exist or does not have an entry in GALIASTABLE, you will not be able to create feature classes in Oracle using GeoMedia. If the table exists, has an entry in GALIASTABLE, and is populated then the geometry fields can be created and indexed. By default, GeoMedia assumes that all geometries will be created in 3D. If you want to utilize just 2D in Oracle, remove the Z_LowerBound, Z_UpperBound, and Z_Tolerance parameters. geometries are created in If Z parameters are absent, 2-D geometries are created

The default coordinate system assigned to a schema is also listed here. The format for default coordinate system entries is as follows:

GPARAMETER: <USER>.DEFAULTCOORDINATESYSTEM

GVALUE: <CSGUID>

GPICKLISTS

The GPICKLISTS metadata table contains the Picklists assignments used by the **Attribute Properties** dialog box in GeoMedia Professional. Also known as domain lists, Picklists allow for a pre-defined list of values to be used when updating attribute fields. GPICKLISTS is defined as follows:

Name	Virtual	Type	Nullable
FEATURENAME	☐	VARCHAR2(64)	☐
FIELDNAME	☐	VARCHAR2(32)	☐
PICKLISTTABLENAME	☐	VARCHAR2(64)	☑
VALUEFIELDNAME	☐	VARCHAR2(32)	☑
DESCRIPTIONFIELDNAME	☐	VARCHAR2(255)	☑
FILTERCLAUSE	☐	VARCHAR2(255)	☑

- FEATURENAME and FIELDNAME combine to form the multi-column primary key.
- FEATURENAME – Contains to the feature class that will use the Picklist. The FEATURENAME should be in the format OWNER.FEATURECLASS.
- FIELDNAME – Contains the specific attribute field in the feature class that will use the Pick List.
- PICKLISTTABLENAME – Specifies a table in the schema containing the Picklist values. This could be a new or an existing feature class.
- VALUEFIELDNAME –The field in the Pick List table that contains the values to be stored in the database. The datatype of the field in the Pick List table specified here must match the datatype of the attribute assigned in the FIELDNAME.
- DESCRIPTIONFIELDNAME – Specifies the field that contains Picklist descriptions to be displayed in the pop-up menu on the **Properties** dialog box. The values stored in VALUEFIELDNAME and DESCRIPTIONFIELDNAME could be the same when the displayed values are the same as the stored values.
- FILTERCLAUSE is optional and may contain an SQL WHERE clause that will be used to filter the records in Picklist. The filter allows a single Pick List table to be used when creating multiple Picklists.

Pick List tables can be any tables that contain the required information, including existing feature classes. You can implement a Picklist as a code list (using separate values and description entries) or as a domain list (when value and description entries are the same). Ranges are not supported. It is up to the DBA to populate the Picklist metadata table with the appropriate entries for the various schemas containing feature classes. A Picklist management tool is available.

GQUEUE and GQUEUEBASE

GQUEUEBASE stores the static queue information used by the **Queued Edit** command in GeoMedia Professional. The GQUEUEBASE table stores information about all static queues used by the schemas in the database instance. The columns in GQUEUEBASE are populated through commands in GeoMedia and are used solely by the **Queued Edit** command. This table is not user editable.

The **Queued Edit** process accesses GQUEUEBASE through a view called GQUEUE that is defined as follows:

```
CREATE OR REPLACE VIEW GQUEUE AS
SELECT QUEUENAME,TABLENAME, QEDITEMDCSFIELDNAME, QEDSTSFIELDNAME,
       QEDCREATOR, QEDNUMGEOMFIELDNAMES, QEDUSEFORMBRGEOMFIELDNAMES,
       QEDADDGEOMFIELDNAMES, QEDNUMSTATUSLIST, QEDSTATUSLISTNAMEVALUE,
       QEDSORTFIELDNAME, QEDSORTASCENDING, READONLYFIELDNAMES,
       NONDISPLAYABLEFIELDNAMES, NONLOCATABLEFIELDNAMES
  FROM GDOSYS.GQUEUEBASE
 WHERE EXISTS
   (SELECT 1 FROM ALL_OBJECTS
     WHERE OBJECT_TYPE = 'TABLE'
       AND OWNER = SUBSTR(TABLENAME,1,INSTR(TABLENAME,'.',1)-1)
AND OBJECT_NAME = SUBSTR(TABLENAME,INSTR(TABLENAME,'.',1)+1))
```

This view ensures that the user sees only the queues for the tables or views they have privilege on. By default, privileges on both GQUEUEBASE and GQUEUE are granted to PUBLIC. The DBA can change this if needed. Any user that needs to create static queues must have SELECT, INSERT, UPDATE and DELETE privileges on both GQUEUEBASE and GQUEUE. Users who review static queues will require SELECT privilege on both GQUEUEBASE and GQUEUE.

LIBRARYTABLES and LIBRARYTABLESBASE

LIBRARYTABLESBASE is similar to the GALIASTABLE and contains a list of the library table types and their associated table names in the format of owner.table. This table is not user editable and the table definition is provided for information only.

Name	Virtual	Type	Nullable
TABLETYPE	☐	VARCHAR2(255)	☑
TABLENAME	☐	VARCHAR2(255)	☐

The **Library** process accesses LIBRARYTABLESBASE through a view called LIBRARYTABLES that is defined as follows:

```
CREATE OR REPLACE VIEW LIBRARYTABLES AS
SELECT TABLETYPE, TABLENAME
  FROM GDOSYS.LIBRARYTABLESBASE
 WHERE EXISTS (SELECT 1
                 FROM all_objects
                WHERE owner = (SELECT user FROM DUAL)
                  AND owner||'.'||object_name = TABLENAME);
```

This view ensures that the user sees only the library tables in the schema they are directly connecting to.

MODIFIEDTABLES

The MODIFIEDTABLES is a required metadata table that lists the tables that are tracked in the MODIFICATIONLOG. As tables are edited, entries are automatically added to the MODIFIEDTABLES table if they do not already exist. The definition of this table is as follows:

Name	Virtual	Type	Nullable
MODIFIEDTABLEID	☐	INTEGER	☐
TABLENAME	☐	VARCHAR2(61)	☑
KEYFIELD1	☐	VARCHAR2(30)	☑
KEYFIELD2	☐	VARCHAR2(30)	☑
KEYFIELD3	☐	VARCHAR2(30)	☑
KEYFIELD4	☐	VARCHAR2(30)	☑
KEYFIELD5	☐	VARCHAR2(30)	☑
KEYFIELD6	☐	VARCHAR2(30)	☑
KEYFIELD7	☐	VARCHAR2(30)	☑
KEYFIELD8	☐	VARCHAR2(30)	☑
KEYFIELD9	☐	VARCHAR2(30)	☑
KEYFIELD10	☐	VARCHAR2(30)	☑
KEYFIELD11	☐	VARCHAR2(30)	☑
KEYFIELD12	☐	VARCHAR2(30)	☑
KEYFIELD13	☐	VARCHAR2(30)	☑
KEYFIELD14	☐	VARCHAR2(30)	☑
KEYFIELD15	☐	VARCHAR2(30)	☑
KEYFIELD16	☐	VARCHAR2(30)	☑

- MODIFIEDTABLEID – This field contains the OBJECT ID for the tables and views that will be tracked in the MODIFICATIONLOG table.
- TABLENAME - Contains the table/view name in the format of OWNER.TABLENAME.
- KEYFIELD1-KEYFIELD16 – KEYFIELD1 contains the primary key identifier for the table (or view). If multi-column primary keys are used, the other KEYFIELDs will store each column making up the primary key.

This table is never cleared and over time, as tables are deleted, may contain orphans. To improve performance, this table can be periodically truncated. However, do not clear this table while there are open GeoMedia Professional sessions. The **Clear Modification Log** command in **Database Utilities** will truncate this table and the MODIFICATIONLOG table.

MODIFICATIONLOG

The MODIFICATIONLOG metadata table tracks modifications made to all GeoMedia feature classes. This required table is used to track all inserts, updates, and deletes made to the tables listed in MODIFIEDTABLES. This table is defined as:

Name	Virtual	Type	Nullable
MODIFICATIONNUMBER	☐	INTEGER	☐
TYPE	☐	VARCHAR2(1)	☑
MODIFIEDTABLEID	☐	INTEGER	☑
SESSIONID	☐	INTEGER	☑
MODIFIEDDATE	☐	DATE	☑
KEYVALUE1	☐	VARCHAR2(255)	☑
KEYVALUE2	☐	VARCHAR2(255)	☑
KEYVALUE3	☐	VARCHAR2(255)	☑
KEYVALUE4	☐	VARCHAR2(255)	☑
KEYVALUE5	☐	VARCHAR2(255)	☑
KEYVALUE6	☐	VARCHAR2(255)	☑
KEYVALUE7	☐	VARCHAR2(255)	☑
KEYVALUE8	☐	VARCHAR2(255)	☑
KEYVALUE9	☐	VARCHAR2(255)	☑
KEYVALUE10	☐	VARCHAR2(255)	☑
KEYVALUE11	☐	VARCHAR2(255)	☑
KEYVALUE12	☐	VARCHAR2(255)	☑
KEYVALUE13	☐	VARCHAR2(255)	☑
KEYVALUE14	☐	VARCHAR2(255)	☑
KEYVALUE15	☐	VARCHAR2(255)	☑
KEYVALUE16	☐	VARCHAR2(255)	☑

- MODIFICATIONNUMBER – Contains values from the GMODLOG sequence and is automatically incremented as edits are made
- TYPE – Contains the type of edit made to the data. The types used are 1 for insert, 2 for update and 3 for delete.
- MODIFIEDTABLEID – Contains the OBJECT ID for the tables and views that will be tracked in the MODIFICATIONLOG table. The MODIFIEDTABLEID is the common link between MODIFICATIONLOG and MODIFIEDTABLES.
- SESSIONID – Contains Oracle session identifier.
- MODIFIEDDATE – This contains the system date when the modification was made.
- KEYFIELD1-KEYFIELD16 – KEYFIELD1 contains the primary key value for the row where the edit has occurred. If a multi-column primary key is used then the other KEYFIELDs will store the values for the other components of the primary key.

IMPORTANT: Because all edits made to all feature classes in the Oracle instance are tracked in the MODIFICATIONLOG table, this table can grow very large very quickly. The size of the MODIFICATIONLOG table can negatively impact editing performance in GeoMedia Professional, so the table should be periodically truncated. However, do not clear this table while there are open GeoMedia Professional sessions.

The **Clear Modification Log** command in **Database Utilities** will truncate both MODIFICATIONLOG and MODIFIEDTABLES. You can also use the following SQL to clear this table:

```
TRUNCATE TABLE GDOSYS.MODIFICATIONLOG;
```

For best results, set up an Oracle job that will automatically truncate the MODIFICATIONLOG table on a periodic basis. Following is an example of this type of job:

```
CREATE OR REPLACE PROCEDURE CLEAR_MODLOG is
BEGIN
 EXECUTE IMMEDIATE 'TRUNCATE TABLE GDOSYS.MODIFICATIONLOG';
 EXECUTE IMMEDIATE 'TRUNCATE TABLE GDOSYS.MODIFIEDTABLES';
END;
/
-- Create the job
BEGIN
 dbms_job.isubmit(1,'CLEAR_MODLOG;',SYSDATE,+1/24,'SYSDATE+1');
END;
/
```

If you do create a job for this process, make sure you schedule it to run when no GeoMedia sessions are active.

Modification logging can be performed with triggers as an alternative to the automatic logging done by GeoMedia Professional. These triggers also allow tracking of changes made outside of GeoMedia Professional. This ensures that GeoMedia Professional sessions are aware of any data changes being made by other non-GeoMedia Professional sessions. Use **Database Utilities** to set up these modification log triggers.

Sequences in GDOSYS

GDOSYS uses three sequences to populate specific fields used by the metadata tables. Never use these sequences for your own purposes. These sequences are as follows:

- GMODLOG – Sequence for the MODIFICATIONNUMBER field in the MODIFICATIONLOG table.
- GAUTONUMBERSEQUENCE – Sequence used to generate a unique identifier when generating sequences for autonumber fields through the **Feature Class Definition** command.
- FIELDLOOKUPINDEXID1 – Sequence for the field INDEXID in the FIELDLOOKUP table.

Triggers in GDOSYS

DELETEMETADATAGMT is the only trigger maintained in GDOSYS. This trigger checks for and deletes the associated metadata entries in GDOSYS whenever a table or column is deleted anywhere in the Oracle database.

Not all metadata tables are cleared by this trigger:

- Any references to the deleted table in MODIFIEDTABLES or MODIFICATIONLOG will still exist.
- The **Clear Modification Log** command in **Database Utilities** should be used to clear these tables of entries.

- The coordinate system used by the deleted table is also retained in the GCOORDSYSTEM table. This is important because tables in this and other schemas may use the same coordinate system as the deleted table.
- Oracle metadata entries are not affected by this trigger. A deleted table will still have entries in USER_SDO_GEOM_METADATA. This can cause a problem if you are creating a new table using the same name as one previously deleted. You must manually clear the entry from Oracle's metadata or define a trigger in the affected schema that will handle this automatically. An example of this trigger is as follows:

```
CREATE or REPLACE TRIGGER DROP_USGM_TRIG
AFTER DROP ON SCHEMA
DECLARE
  v_EXIST INTEGER;
BEGIN
  SELECT COUNT(1) into v_EXIST
    FROM USER_SDO_GEOM_METADATA
   WHERE TABLE_NAME = sys.dictionary_obj_name;

  IF v_EXIST >0 THEN
    DELETE FROM USER_SDO_GEOM_METADATA
     WHERE TABLE_NAME = sys.dictionary_obj_name;
  END IF;

END;
/
```

Metadata Table Relationships

The metadata tables in the GDOSYS schema are related to each other in specific ways. Direct relationships are listed as follows:

- GCOORDSYSTEM.CSGUID = GFIELDMAPPING.CSGUID = GEOMETRYPROPERTIES.GCOORDSYSTEMGUID
- GEOMETRYPROPERTIES.INDEXID = ATTRIBUTEPROPERTIES.INDEXID = FIELDLOOKUP.INDEXID
- FIELDLOOKUP.FEATURENAME = GFEATURES.FEATURENAME
- MODIFIEDTABLES.MODIFIEDTABLEID= MODIFICATIONLOG.MODIFIEDTABLEID

The following are indirect and implied relationships:

- The OWNER/TABLE_NAME in the GFIELDMAPPING table corresponds to the FEATURENAME column in the client metadata tables (FIELDLOOKUP and GFEATURES).
- The COLUMN_NAME in the GFIELDMAPPING table corresponds to the FIELDNAME column in other metadata tables.

Exercise caution when making direct edits to the metadata tables in GDOSYS. If you do not include all the required information, you may get strange results in the GeoMedia environment. For best results, any metadata changes should be made using **Database Utilities**.

USER ACCOUNTS AND PRIVILEGES

As a DBA, you have a lot of latitude in the creation of user accounts in Oracle. What privileges you assign to the user account is dependent on what purpose the account will serve. A typical user account is created using the following syntax:

```
CREATE USER <user> PROFILE "DEFAULT"
    IDENTIFIED BY <pswd>
    DEFAULT TABLESPACE USERS
    TEMPORARY TABLESPACE TEMP
    ACCOUNT UNLOCK;
```

where <user> is the username you want to create and <pswd> is the password for that user account. If this is the account that will contain the spatial data, tables, views and other objects, you will need to assign specific privileges to give the account that capability. The following privileges are just an example:

```
GRANT UNLIMITED TABLESPACE TO <user>;
GRANT CONNECT TO <user>;
GRANT RESOURCE TO <user>;
GRANT CREATE TABLE TO <user>;
GRANT CREATE SEQUENCE TO <user>;
GRANT MERGE ANY VIEW TO <user>;
```

- The UNLIMITED TABLESPACE role is optional but can be useful when creating and populating spatial tables.
- The CONNECT and RESOURCE roles are common and are required if your user needs to create database objects.
- The CREATE TABLE and CREATE SEQUENCE privileges are required in order to use spatially related PL/SQL functions and procedures. These must be granted explicitly and cannot be granted as part of a role.
- The MERGE ANY VIEW privilege is required if you are using Oracle 10.2 or later and it also must be granted explicitly. MERGE ANY VIEW allows the optimizer to improve query performance and is essential when dealing with spatial data. You can also handle this at the database level by setting the initialization parameter OPTIMIZER_SECURE_VIEW_MERGING to FALSE.

Two main rules apply to user account creation:

- Never create a user account in the SYSTEM Tablespace. Typically, a user is created in the USER tablespace with temporary storage in TEMP. The actual tablespace used is up to the DBA.
- Never assign the DBA role to a master or satellite user account if you plan to use it with GeoMedia. The DBA role can have serious performance implications for GeoMedia connections.

Satellite user accounts generally contain no database objects. They exist to allow a specific user varying degrees of privilege on the master schema, which hold the database objects (tables, views, and so forth.). All users that need to access a spatial schema need to have, at a minimum, SELECT privilege

on all the objects in both GDOSYS and in the master schema. In addition to SELECT, here are some typical configurations and the required privileges:

- Master User and Schema – Schema objects can only be created by master user.
- Admin User – Has full control of master schema.
  - GRANT ALL on all database objects in GDOSYS.
  - GRANT ALL privileges on the master schema.
- Viewing User – Has read-only access to all objects.
  - SELECT on all GDOSYS objects.
  - SELECT on all master schema objects.
- Editing User – Has edit capability on some feature classes but not others and cannot remove or delete data. Cannot create objects.
  - Grant INSERT and UPDATE on GDOSYS's MODIFIEDTABLES AND MODIFICATIONLOG
  - Grant INSERT and UPDATE on GDOSYS's GQUEUE AND GQUEUEBASE so the user can create and modify static queues in GeoMedia.
  - INSERT and UPDATE on all read-write tables and sequences in master schema. Add DELETE for the ability to delete data.

DATABASE OBJECTS

Tables can be created using GeoMedia Professional's **Feature Class Definition** or using native Oracle commands. The benefit of using **Feature Class Definition** is that the maintenance of the metadata is handled transparently. However, data types assigned via **Feature Class Definition** have to be converted to native Oracle data types. This is not the case if you create tables directly in Oracle. Using SQL, you can create spatial tables just like standard tables; the difference is that you include a column using the spatial datatype. Here is an example of creating a table called STATES that contains two separate geometry fields.

```
CREATE TABLE STATES (
          PID NUMBER(38) NOT NULL PRIMARY KEY,
          ST_NAME VARCHAR2(64),
          GEOM1 MDSYS.SDO_GEOMETRY,
          GEOM2 MDSYS.SDO_GEOMETRY);
```

When creating tables, make sure you use a primary key column(s). Primary keys are required for read-write access and will also improve performance.

If you have existing tables, you can add a field to hold the geometry using Oracle's ALTER table command:

```
ALTER TABLE STATES ADD (GEOMETRY MDSYS.SDO_GEOMETRY);
```

If you create tables outside of GeoMedia Professional, you will need to assign the required metadata for both Oracle and for GeoMedia. You will also need to spatially index your geometry fields.

Non-spatial attribute only tables can also be created and used. In this case, GeoMedia still requires metadata but Oracle does not.

Table and column names must always be expressed in upper case. Mixed case and lower case names are not allowed. When using **Feature Class Definition**, table names are restricted to 24 characters. If you create the tables directly in Oracle, you can use the full 30 character maximum allowed by Oracle. Keep in mind that other applications may also impose length restrictions.

Default Values

Default values can simplify data entry and supply values for columns that are either required or just need to have a specific entry. Default values are honored by GeoMedia but not directly. When inserting a new record with the option to display the **Attribute Properties** dialog box turned on, the default values are not shown in the dialog box even though they are available at the database level. They will be used when the insert occurs. If the fields are required, you will not see an error, instead, the insert will pick up the default values. However, if the option **Copy Attribute Values from Previous Feature** is enabled, you will no longer be able to use the default value. Instead, the value used in the previous insert will be used. If you delete the previous value used in a required field, the default value will still not be used and you will get an error message.

For best results with defaults, either turn off the **Copy Attribute Values from Previous Feature** option or do not make the fields required. Functional based defaults will work but again, you must turn off **Copy Attribute** option. This same problem will occur if you are using triggers to populate required fields.

Views and Join-Views

GeoMedia Professional makes no distinction between views and tables; it treats a view just like any other feature class. The Oracle Object Model data server handles read-write access to most types of views. Names of views are always expressed in uppercase. For views to be updatable in GeoMedia Professional, they must come from a table that is key-preserved. Only key-preserved tables can be updated through a view. One way to ensure this is true is to include a primary key as part of the view definition, which preserves the key's status in the view. In a join-view, the primary key can come from either the left or the right side as long as it retains its key status. It cannot come from both sides of the join and still be updatable. In addition, only the side of the join containing the key will be updatable.

To determine whether a view/join-view is updatable or not, you can check the Oracle view USER_UPDATABLE_COLUMNS:

```
SELECT * FROM USER_UPDATABLE_COLUMNS
  WHERE TABLE_NAME='view_name';
```

Views can be added to the GDOSYS metadata using **Database Utilities**. As with tables, views that contain a column of type SDO_GEOMETRY must also have an entry in USER_SDO_GEOM_METADATA (Oracle does not require this, but GeoMedia does). All references to views in GALIASTABLE, MODIFIEDTABLES, GFEATURES, and FIELDLOOKUP will be expressed as OWNER.VIEW.

The following three steps are required to make a view compatible with GeoMedia Professional:

1. Creating a View – Create a view in Oracle. For read-write capabilities, the view definition must contain a primary key column from the underlying base table. Multi-column primary keys are allowed.
2. Update USER_SDO_GEOM_METADATA – All views that contain a column of type SDO_GEOMETRY need to have an entry in Oracle's spatial metadata view USER_SDO_GEOM_METADATA.
3. Run GeoMedia Professional's **Database Utilities** – The Oracle Object Model data server treats Oracle views as standard GeoMedia feature classes. The views are handled in a special way through **Database Utilities**.
   While inserting feature class metadata, the **Attributes** tab of the feature class **Properties** dialog box of the view will have a special button for selecting the primary key for the view. The column selected must be unique, not null, and should be the same as the key column in the base table. This pseudo-key column makes the view key preserved. **Database Utilities** will populate the GDOSYS.GINDEXCOLUMNS table with the necessary information to make the view read-write inside GeoMedia Professional.
   For insert operations, GeoMedia must populate the primary key value itself; it cannot be populated through a trigger. To force this, assign the datatype for the column selected to be the key (in Step 3) to autonumber, and pick the sequence associated with the same column in the base table.

Modifiable join-views are the most difficult to work with. In this case, the primary and foreign keys should be explicitly defined in the underlying base tables. The concept of key-preserved tables is fundamental to understanding the restrictions on modifying join views. For more information, refer to Oracle's documentation on modifiable join-views. If you need to have full edit capability on your join views, you need to make use of 'INSTEAD OF' triggers.

The following is an example of a join-view that has a join relation between a graphic feature class (PARCEL_GEOM) and a non-graphic feature class (PARCEL_INFO):

```
CREATE VIEW PARCEL_GEOM_AREA_INFO AS
  SELECT G.GEOMETRY, G.PARCEL_ID, G.AREA, G.PID, P.LOCNO,
         P.RECTYPE, P.YEARBUILT, P.API_VALUE1, P.LANDUSE_CODE
    FROM PARCEL_GEOM G, PARCEL_INFO P
   WHERE G.PID = P.PID
     AND G.AREA >= 100000
     AND G.AREA <= 200000;
```

If you plan to edit join-views, you should ensure that the autonumber sequence assigned to the key on the base table key is also assigned to the same key used in the view. This will ensure that the base table sequence remains in sync. Also, GeoMedia must be able to populate this pseudo key field in order to perform inserts. This can be done manually or by changing the datatype of the key field to autonumber and using the sequence assigned to the base table. Use **Database Utilities** for sequence assignments.

When you inset, update, or delete a record from a view, the underlying base table is modified by Oracle. If you are displaying both the base table and the view in the GeoMedia map window, only the view will be updated. This happens because Oracle, not GeoMedia Professional, is updating the base table. For notification of the change to reach GeoMedia Professional, you will need to set up Modification Log Triggers on the base table. These triggers ensure that changes to the base table made outside of GeoMedia are reflected inside of GeoMedia Professional without having to close and re-open the database connection. Manually adding the view's information to the Modification Log Trigger of the base table will also ensure proper notification for the view when the base table is updated. Modification Log Triggers are optional and can be created using **Database Utilities**.

SEE "MODIFICATION LOGGING" IN THIS APPENDIX FOR MORE INFORMATION.

Materialized views can be used through the OOM server to increase the speed of queries, especially native queries, on very large datasets. Rewriting queries to use materialized views rather than detail tables (feature classes) greatly improves response time. However, materialized views are treated as snapshots (by both Oracle and GeoMedia) and are considered read-only. An example of a materialized view is the following:

```
CREATE MATERIALIZED VIEW CITIES_INSIDE_STATES AS
  SELECT A.STATE_NAME, B.GEOMETRY, A.ID, B.CITY_NAME
    FROM STATES A, CITIES B
   WHERE SDO_RELATE(B.GEOMETRY,A.GEOMETRY,'MASK=INSIDE
```

```
    QUERYTYPE= JOIN')='TRUE'
AND A.STATE_NAME LIKE 'A%';
```

Triggers

Database triggers can be an extremely useful tool to manage and control data and to enforce data rules and logic. Some of the more commonly used areas include logging, DML operations, and the enforcement of data rules.

Modification Logging

GeoMedia Professional automatically records insert, update, and delete operations on features in the GDOSYS's MODIFICATIONLOG table. Changes to the schema made outside GeoMedia Professional are not recorded. To make external changes visible to GeoMedia Professional while in a session, you need to set up modification triggers. These triggers are assigned to table-based feature classes using the GeoMedia Professional **Database Utilities**. The trigger name will be <FEATURE_CLASS>GMT. For example, the trigger on STATES would be STATESGMT. A single trigger handles the insert, update, and delete operations.

If GeoMedia Professional detects the presence of the modification log triggers, it will use them rather than its own internal logging. The use of triggers may improve performance in editing operations because modification logging is being done by the server.

Views pose a different kind of problem because normal triggers cannot be assigned to views. When a view is edited, Oracle is actually editing the underlying base table. GeoMedia will automatically log modifications for the view, but will not show an update for the base table unless a modification log trigger is created on the base table.

Another issue arises when both the view and the base table are displayed on the legend in GeoMedia. If a change is made to the view and a modification log trigger is active on the base table, updates to the view will also cause an update on the base table. The changes will be immediately visible in the map window. If the base table is updated, the view will also be updated at the database level. GeoMedia will not be aware of that change and will not show it in the map window unless the warehouse is closed and re-opened. There are two ways to avoid this issue:

- You can add entries for the view to the base table's modification log trigger. These can be quite complicated to set up and requires knowledge of PL/SQL. They can also lead to a redundant entry for the view in the MODIFICATIONLOG table because both GeoMedia and the trigger will add an entry for the view.
- The best solution here is to create another trigger on the base table that uses the name of the view. For example, the trigger for STATES_VIEW would be called STATES_VIEWGMT. The trigger would fire for the base table STATES but would update the MODIFCATIONLOG table for the view. The trigger itself would be nearly identical to the base table's trigger STATESGMT, which would also fire. To do this, make a copy of the trigger used for the base table, rename it to reflect the view's name, and then change the references to the base table in the body of the trigger to reference the view instead.

Sequences

Another important use of triggers is to automatically utilize the sequence when using an integer-based primary key. GeoMedia will use the sequence automatically if the metadata for the key field is set to AUTONUMBER. If you edit the table outside of GeoMedia, the sequence will not be automatically used unless you create a trigger to handle it. An example of this type of trigger for a feature class called STATES is shown below:

```
CREATE OR REPLACE TRIGGER STATES_INSERT_TRIG
BEFORE INSERT ON STATES
REFERENCING OLD AS OLD NEW AS NEW FOR EACH ROW
  IF :NEW.ID IS NULL THEN
    SELECT STATES_ID_SEQ.NextVal
   INTO :NEW.ID FROM dual;
  END IF;
END;
/
```

You can assign sequence for use with any numeric field, but only one field can have the distinction of being autonumber.

INSTEAD OF Triggers

INSTEAD OF triggers are special types of trigger used to maintain views, specifically join views. These triggers are nothing more than a PL/SQL procedure that uses the columns of the view as the input parameters. Using this type of trigger allows you to edit join views by managing the insert, update, and delete operations for the underlying base tables. This is the only way to have full read write access to join views.

Remember, for inserts to be successful, GeoMedia must populate the primary key automatically. If the 'INSTEAD OF' trigger populates the primary key, the insert will fail. Using GeoMedia Professional's **Database Utilities**, change the datatype for the column in that is the pseudo key to autonumber and assign the base table's sequence to the column.

When modifying the data in a join view using GeoMedia Professional, the software will first attempt to write directly to the view. If an INSTEAD OF trigger is available, the trigger will process the insert, update, or delete operation. If a trigger is not available, Oracle will attempt to write directly to the underlying tables. This may return an error result from Oracle. If you are using triggers to populate required fields, make sure you turn off the **Copy Attribute Values from Previous Feature** option.

DATABASE UTILITIES

The **Database Utilities** tool consists of several utilities for managing and updating GeoMedia metadata in Access, Oracle, and SQL Server databases. These utilities are delivered with GeoMedia Professional and GeoMedia Web Map. These are DBA utilities and should not be used by normal database users.

SEE THE DATABASE UTILITIES ONLINE HELP FOR COMPLETE INFORMATION ON THESE utilities.

You can access **Database Utilities** from **Start > Programs > GeoMedia Professional > Utilities > Database Utilities**.

To use Database Utilities to create the GDOSYS schema, you will need to log in to Oracle as a DBA or system/manager to create and to populate the necessary GeoMedia metadata. For operations involving a specific schema, you can log in as the user who owns the schema if that user has full privileges on the GDOSYS metadata schema.

If the GDOSYS schema does not exist in your Oracle database, you will need to create it before any of the commands will work. Use the **Create Metadata Tables** button to create and populate the GDOSYS schema. You will need to be connected as DBA for this to work correctly. This command is useful in two ways: 1) It creates metadata tables for native databases, and 2) it checks and repairs sparse metadata or updates metadata for new releases. Once the GDOSYS schema is created, you will not be prompted to create it again.

Using an Existing Oracle Spatial Schema

If you already have data in an Oracle schema that you want to utilize within the GeoMedia environment, you need to ensure the following:

- The Oracle 10g Client software has been loaded, and a host has been configured using Oracle's **Network Configuration Utility**.
- The database schema resides in Oracle 9.2 or later.
- USER_SDO_GEOM_METADATA is populated with the extents of the data stored in all SDO_GEOMETRY columns in the schema.
- An RTree index has been created for every geometry field.

For read-only connections, you will also need to ensure that:

- The GeoWorkspace coordinate system used to view the data is in the same coordinate system as the data.

For read-write connections, you will also need to ensure that:

- The metadata schema, GDOSYS, has been created in the database instance, and appropriate privileges have been assigned.
- All the tables in the schema that that will be used in GeoMedia have the appropriate metadata entries in GDOSYS, including a coordinate system. Metadata is required for both graphic and non-graphic tables. Adding the appropriate entries and the associated coordinate systems can be done using **Database Utilities**.

Here is an example of the steps required to make an existing schema available in the GeoMedia environment. In this example, you have an existing spatial schema in Oracle called USA_DATA containing two feature classes, STATES and COUNTIES.

For read-only connections to USA_DATA, you need to ensure that the Oracle metadata view USER_SDO_GEOM_METADATA has been populated with the correct values. Use the following SQL to see if there are existing entries:

```
SELECT * FROM USER_SDO_GEOM_METADATA;
```

If there are no entries, you will need to create them. The simplest way is to just insert the default range:

```
INSERT INTO USER_SDO_GEOM_METADATA VALUES
  ('STATES','GEOMETRY', MDSYS.SDO_DIM_ARRAY (
  MDSYS.SDO_DIM_ELEMENT('X',-2147483648,2147483647,0.00005),
  MDSYS.SDO_DIM_ELEMENT('Y',-2147483648,2147483647,0.00005),
  MDSYS.SDO_DIM_ELEMENT('Z',-2147483648,2147483647,0.00005)),
  NULL);

INSERT INTO USER_SDO_GEOM_METADATA VALUES
  ('COUNTIES,'GEOMETRY', MDSYS.SDO_DIM_ARRAY (
  MDSYS.SDO_DIM_ELEMENT('X',-2147483648,2147483647,0.00005),
  MDSYS.SDO_DIM_ELEMENT('Y',-2147483648,2147483647,0.00005),
  MDSYS.SDO_DIM_ELEMENT('Z',-2147483648,2147483647,0.00005)),
  NULL);

COMMIT;
```

The next step in preparing data is to create the spatial indexes that are required for spatial filters. The easiest index to create is an RTree index, and the syntax is shown below:

```
CREATE INDEX STATES_RT ON STATES(GEOMETRY)
    INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX COUNTIES_RT ON COUNTIES(GEOMETRY)
    INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

These feature classes are now ready for display in GeoMedia Professional through a read-only connection. Because the coordinate system is not known, you will need to ensure that the workspace contains the correct coordinate system for your data. Features will be displayed as a SPATIALANY or a Compound data type.

For a read-write connection, GeoMedia metadata must be present and populated with the correct values. This requires the existence of the GDOSYS metadata schema. Create and populate the GDOSYS schema using the following steps:

1. Open **Database Utilities** and connect to your Oracle instance as SYSTEM/MANAGER (or any DBA user).
2. Click **Create Metadata Tables**.

GDOSYS should be created.

3. Once GDOSYS has been created, click **Change** to connect the schema containing your data (in this case, USA_DATA).
4. On the **Database Utilities** dialog box, click **Insert Feature Class Metadata**.

A list of available features is listed on the left side of the dialog box.

5. Select the features to add by moving them to the right side of the dialog box.
6. Select each feature class on the right, and then click **Properties**.
7. If you need to assign/create a sequence for a primary key or other column, select the **Attributes** tab; then set the data type of the column you want to sequence to **Autonumber**.

You are prompted to select and assign a sequence to that column.

8. Select an existing sequence, or create and assign a new sequence. Sequences can only be assigned to integer data types.
9. On the **Feature Class Properties** dialog box, select the **Geometry** tab, and assign the correct geometry type to the **Geometry** column using the spatial data types wherever possible.
10. Continue assigning the appropriate properties to each feature you want to add.
11. When finished, click **OK** on the **Insert Feature Class Metadata** dialog box.

You are prompted to assign a coordinate system to the list of features.

12. Pick an existing coordinate system, or create and assign a new coordinate system. Optionally, set the coordinate system as the default coordinate system for the schema.

After metadata has been assigned, you can establish read-write connections to the schema through GeoMedia.

Creating a New GeoMedia Warehouse in Oracle

To create a new warehouse in Oracle for GeoMedia to use, you will first need to create the GDOSYS schema. This is outlined above in Steps 1-3 and requires DBA privileges on the database. After GDOSYS is created, your DBA needs to create a user account in Oracle that will hold the spatial object tables. Then use one of the following to load and use the data:

- Use Oracle statements to construct the table definitions and to insert the required data. Once the schema is defined and loaded, you will need to populate the Oracle metadata USER_SDO_GEOM_METADATA, index the data, and then use the **Database Utilities** to populate the required metadata in the GDOSYS schema (for read-write access).
- Use the GeoMedia Professional **Output to Feature Classes** command to import table definitions and data from other data warehouses, for example, ArcView or MGE. In this case, Oracle metadata, indexes, and the GDOSYS metadata are populated for you. The spatial indexes will not be optimal, so re-indexing will probably be required.
- Use the GeoMedia Professional **Export to Oracle Object Model** command to output a set of SQL Loader files that will construct the database tables and bulk load the data. In this case, both Oracle metadata and the GDOSYS metadata are populated for you. The data will need to be spatially indexed.
- Create new feature classes using the GeoMedia Professional **Feature Class Definition** command, and populate the data by digitizing new elements in the map window. In this case, Oracle metadata,

indexes, and the GDOSYS metadata are populated for you. Because the default index is RTree and the newly created table is empty, the table will need re-indexing once data is loaded.

APPENDIX A : DATA SERVER EXTENSIONS TO THE ORACLE SPATIAL DATA MODEL

The GeoMedia product line supports geometries as described in the GDO (Geographic Data Objects) specification, available separately. The binary structure of these geometry BLOBs is in the public domain, and represents the uniform currency in GeoMedia for conveying geometric information. When a GDO data server conveys geometries to a GeoMedia product, it does so by converting the storage structure of geometries in the native format of its warehouse, into GDO geometry BLOB format.

In the case of a read-write GDO data server, that native-to-GDO format conversion must also be supported in the reverse, a GDO-to-native format conversion. For the Access and SQL Server data servers, there is no "native" storage format for geometry, so the GDO geometry BLOBs are stored directly as binary data. However, in the case of the Oracle Object Model (OOM) data server, a conversion must take place from the GDO geometry BLOB format into the native geometry store of Oracle.

For most GDO geometry types this is a straightforward endeavor as there are structures in parallel between the two. However, in several cases the GDO geometry is richer than the corresponding Oracle geometry, or there simply is no corresponding Oracle geometry, thus causing data loss if restricted to a straight mapping.

To overcome this problem, the OOM data server uses Oracle's ability to include user element data with the Oracle geometry elements via an EType of 0 (zero). In each case, the base Oracle geometry type is preceded by a GeoMedia-specific descriptor structured as follows:

```
EType:  0 (zero)
Interpretation:  <varies according to GDO geometry type>
Ordinate Array:  <varies according to Interpretation>
```

By Oracle convention, EType values of 0 (zero) indicate an application-defined element, which is ignored by Oracle's spatial functions.

The nature and ownership of any given application-defined element is driven by the interpretation value, which is unique across all Oracle-based applications. Interpretation numbers are managed by Oracle, which maintains a clearinghouse of such numbers in order to ensure that conflicts do not occur.

The ordinate array is a variable-length sequence of double-precision floating point numbers, but can be coerced to hold other types of data[1]. The contents of the ordinate array are dictated by the interpretation value.

This technique is used as described below to faithfully model oriented point, text point, and raster geometries.

---

1 Unfortunately, Oracle strictly interprets these values as double-precision floating point numbers, which means that care must be taken in how text and integer data is packed into them. It is not always possible to fill all 8 bytes of each array member with data.

TextPointGeometry

Oracle does not have a text geometry type. To model this geometry, the Oracle Object Model data server must embellish a point geometry. It does so as follows:

```
Base Oracle type:    Point geometry
GType: 1
EType: 1
Interpretation:1

Preceding user-defined element:
EType: 0
Interpretation:6001
Ordinate Array (for EType 0):
```

Rotation
Normal Vector (i)
Normal Vector (j)
Normal Vector (k)
Options/Format/Alignment
Text Size
Text (bytes 1-4)
Text (bytes 5-8)
<repeating…>

The **rotation** value is a double-precision floating point number representing the rotation around the normal unit vector, in radians.

The **normal vector** values are 3 double-precision floating point numbers representing the i,j,k values of the normal unit vector of the text. Each of the values must be in the range 0 (zero) to 1 (one).

The **options/format/alignment** value is a long integer value stored as a double-precision floating point number, representing the options, text format and alignment values of the text.

The **alignment** portion is stored in the lowest-order byte of the long integer. It may take on the following values:

```
0 – center center
1 – center left
2 – center right
4 – top center
5 – top left
6 – top right
8 – bottom center
9 – bottom left
10 – bottom right
```

The **format** portion is stored in the second-lowest-order byte of the long integer. It may take on the following values:

```
0 – rich text (encoded in the local code page per RTF specification)
1 – unformatted text (encoded in the local code page)
2 – Unicode text (UTF-16 encoding)
```

The options portion is stored in the third-lowest-order byte of the long integer. It is a bitmask that indicates the settings of various options. A single option is currently defined, as follows:

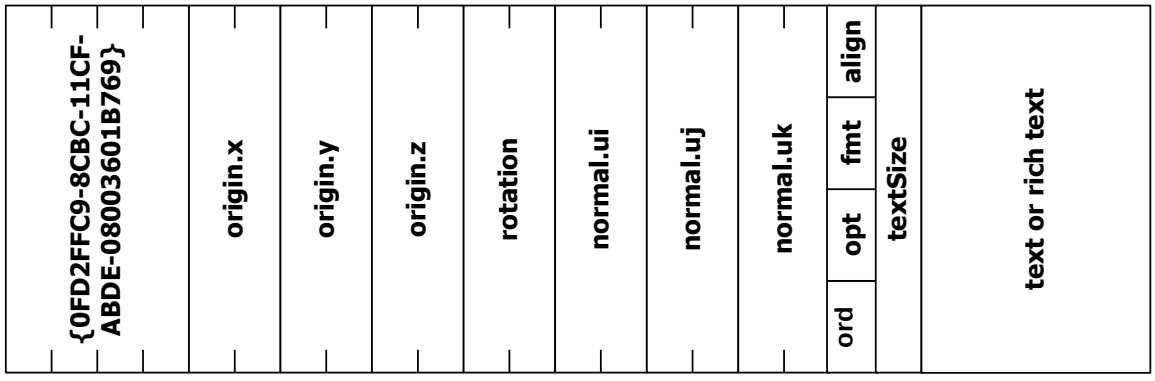
0x01 – composite text indicator. This option may be set only if the geometry is part of a Collection Geometry containing an ordered set of single-character TextPointGeometry members exclusively. All members must have the same setting for this option. If set, it indicates that, whenever necessary or appropriate (e.g.: for purposes of editing the string), the characters of the individual members are to be concatenated and treated as a single word or phrase rather than as logically independent and distinct text strings.

The **text size** value is a long integer value stored as a double-precision floating point number, representing the number of bytes in the text string.

The **text** value is a variable number of double values representing the text characters of the text object, in multi-byte format, including any embedded rich text qualifiers. Character data is packed, 4 bytes per double value, by embedding the bytes in an unsigned long integer variable and storing this in the double value. The character data is packed with the first, fifth, ninth, etc. bytes in the highest-order byte of the long integer, and the fourth, eighth, twelfth, etc. bytes in the lowest-order byte.

Text Point Geometry GDO

Blob Format



Discussion

Size = textSize + 80 bytes

The Text Point Geometry consists of the GUID, followed by the origin point, the rotation relative to the X axis, the normal vector, a properties word containing the order, options, format and alignment, the size of the text, and the text characters themselves.

The vector is normal to the plane in which the text lies.

The order field (ord) indicates the reading order of the whole text. As the text can consist of subtexts with different reading orders, this field's value determines the layout of the subtexts in relationship to each other. Possible values for the ord field are:

- gptReadingOrderUndefined=0x00; this value indicates that the reading order of the text is not defined. The reading order is determined by the application. For GeoMedia products, this means the reading order used through version 6.1.7 of the GeoMedia desktop core products, which is as follows. If the text has attributes that require complex script processing, the reading order is set to right-to-left. (Some examples of the attributes are bidirectional rendering, contextual shaping and combining characters.) Otherwise, the reading order is set to left-to-right.
- gptReadingOrderLTR=0x01; this value indicates that the reading order is left-to-right. Most languages use this order. When the order field is set to gptReadingOrderLTR, the subtexts are laid out from left to right.
- gptReadingOrderRTL=0x02; this value indicates that the reading order is right-to-left. Some languages (e.g. Arabic and Hebrew) use this order. When the order field is set to gptReadingOrderRTL, the subtexts are laid out from right to left.

The options field (opt) is a bit mask that indicates the settings of various options. A single option is currently defined:

- gptCompositeText = 0x01; This option may be set only if the geometry is part of a Collection Geometry containing an ordered set of single-character TextPointGeometry members exclusively. All members must have the same setting for this option. If set, it indicates that, whenever necessary or appropriate (e.g.: for purposes of editing the string), the characters of the individual members are to be concatenated and treated as a single word or phrase rather than as logically independent and distinct text strings.

All unassigned bits in the options field bit mask must be clear.

The format (fmt) indicates whether the characters are stored as rich text, multi-byte character strings cast in the local code page, or Unicode using the UTF-16 encoding. If rich text, then the characters may be either MBCS (local code page) or Unicode (UTF-16) depending on the rich text formatting options (see note below). Possible values for the fmt field are:

- gmfRichText = 0x00;
- gmfMultiByte = 0x01;
- gmfUnicode = 0x02

Alignment (align) indicates the location of the origin point relative to the bounding box of the text. Possible values are:

- cgmaTopLeft = 0x05;
- cgmaTopCenter = 0x04;
- cgmaTopRight = 0x06;
- cgmaCenterLeft = 0x01;
- cgmaCenterCenter = 0x00;
- cgmaCenterRight = 0x02;
- cgmaBottomLeft = 0x09;
- cgmaBottomCenter = 0x08;
- cgmaBottomRight = 0x0a;

The text size is expressed in bytes – not characters – regardless of the format. The minimum size is zero. There is no logical limit on the maximum size.

Not all clients support all of the features of rich text. Commonly supported features include:

- MBCS/Unicode flag
- Font characteristics (font name, bold, underline, italic, color, size).

RasterGeometry

Starting with Oracle version 10g, Oracle has a native Raster Geometry format (SDO_GEORASTER) that is supported with the 6.0 HF#6 (06.00.34.84) version of the Oracle Object Model Data Server. With this enhancement, the data server now supports two types of raster storage:

- Conventional GeoMedia storage using an extended Oracle polygon geometry referencing either a fully qualified raster filename in UNC notation, or an XML document identifying an alternative raster data source.
- New Oracle GeoRaster storage embedded in the database.

Regardless of the storage technique, the data server presents the data as a RasterGeometry. The sections below describe the storage (conventional technique only) and presentation (both techniques) of these types.

Conventional GeoMedia Storage

The conventional GeoMedia storage technique is read-write. The data server performs the modeling of this geometry by embellishing a polygon geometry as follows:

Base Oracle type: Polygon geometry[2]

GType:	3
EType:	1003
Interpretation:	1

Preceding user-defined element:

EType:	0
Interpretation:	6002

Ordinate Array (for EType 0):

Display Matrix (1)
Display Matrix (2)
…
Display Matrix (16)
Moniker Size
Moniker (char 1-4)
Moniker (char 5-8)
<repeating…>

The **display matrix** value is 16 consecutive doubles representing the 4x4 display matrix, in the order given below. The structure and contents of the matrix are as follows:

1. **Scale X**	5. **0.0**	9. **0.0**	13. **0.0**
2. **0.0**	6. **Scale Y**	10. **0.0**	14. **0.0**
3. **0.0**	7. **0.0**	11. **Scale Z**	15. **0.0**

[2] The raster outline is a 4 point polygon. The first/last point is repeated so this geometry contains 5 points.

4. **Delta X**	8. **Delta Y**	12. **Delta Z**	16. **0.0**

The **moniker size** value is a long integer value stored as a double-precision floating point number, representing the number of wide characters in the text string.

The **moniker** value is a variable number of double values representing the characters of the moniker referencing the raster data, in wide character format (UTF-16 encoding). Characters are packed, 2 wide characters per double value, by embedding the characters in an unsigned long integer variable and storing this in the double value. The characters are packed with the first, third, fifth, etc. wide characters in the two lowest-order bytes of the long integer, and the second, fourth, sixth, etc. wide characters in the two highest-order bytes.

The moniker contains either the fully qualified raster filename in UNC notation, or an XML document identifying an alternative raster data source.

Oracle GeoRaster Storage

The Oracle GeoRaster storage technique is read-only. The data server uses the native GeoRaster storage to compute the footprint polygon and display matrix. It then populates the moniker with an XML document. The XML provides information sufficient to query the Oracle database for the raster data. This is referred to as an Oracle GeoRaster (OGR) moniker. Because the XML schema could change over time in response to customer requirements, it is recommended that a generic XML parser be used to extract the items/elements from the string.

Example XML:

```
<?xml
version="1.0"?><OGRMoniker><DBName>valdb</DBName><EnvHandle>5b005a8</EnvHandle>
<SvcHandle>5b276bc</SvcHandle><TableName>NALA_IMAGES</TableName><ColumnName>G
EORASTER</ColumnName><RDTName>RDT_111$</RDTName><RasterID>112</RasterID><Con
nectString>RAS_NALA/*@valdb</ConnectString></OGRMoniker>
```

Description of XML elements:

Element	Description
<?xml version="1.0"?>	Identifies the string as XML. There are other data servers in GeoMedia that use XML strings for other raster data sources, so the presence of this element does not by itself indicate an OGR Moniker from the Oracle Object Model data server.
<OGRMoniker>	Signals the presence of an OGR moniker.
<DBName>	Contains the name of the database 'service' in Oracle terminology.
<EnvHandle>	For Intergraph internal use only.
<SvcHandle>	For Intergraph internal use only.
<TableName>	Contains the name of the table which has the SDO_GEORASTER column, expressed in <OWNER.TABLE> form.
<ColumnName>	Contains the name of the SDO_GEORASTER column within the table.
<RDTName>	Contains the name of the associated RasterDataTable. Redundant with information within the SDO_GEORASTER of the column identified above, but provided for convenience.
<RasterID>	Contains the ID within the raster data table, of the item in the column "ColumnName" of the table "TableName" within the service "DBName". Identifies a unique row from the table, for loading a particular GeoRaster. Also is a key in the raster data table for each tile/line (blob) of data.
<ConnectString>	Contains a password-hidden version of the connect string for Oracle. When making a connection to the database, this information can be used to pre-populate a login dialog so the user only need enter the password.