GeoServer Database Research

Submitted To: Program Manager
GeoConnections
Victoria, BC, Canada

Submitted By: Jody Garnett
Brent Owens
Refractions Research Inc.
Suite 400, 1207 Douglas Street
Victoria, BC, V8W-2E7
jgarnett@refractions.net
Phone: (250) 885-0632
Fax: (250) 383-2140

TABLE OF CONTENTS

GEOSERVER DATABASE RESEARCH 1

TABLE OF CONTENTS 2

1 INTRODUCTION 3

2 WEB FEATURE SERVER REQUIREMENTS 4

2.1 WEB FEATURE SERVER OPERATIONS 4
2.2 TRANSACTION WEB FEATURE SERVER OPERATIONS 5
2.3 WEB FEATURE SERVER LOCKING 6
2.4 DATA MODELING 7
2.5 GEOGRAPHIC MARKUP LANGUAGE 7
2.6 REFERENCES 11

3 GEOSERVER DATA ACCESS 12

3.1 GEOTOOLS2 DATA ACCESS 13
3.2 GEOSERVER FEATURE TYPE ACCESS 15

4 DATABASE COMPARISON 16

5 POSTGIS 17

5.1 POSTGRESQL JDBC DRIVERS 17
5.2 POSTGIS DATA MODELING 18
5.3 POSTGRESQL LOCKING SUPPORT 19
5.4 POSTGIS DATASOURCE 20
5.5 POSTGIS REFERENCES 20

6 ORACLE SPATIAL 21

6.1 ORACLE SPATIAL JDBC DRIVERS 21
6.2 ORACLE SPATIAL DATA MODELING 22
6.3 ORACLE SPATIAL LOCKING SUPPORT 23
6.4 ORACLE SPATIAL DATASOURCE 24
6.5 ORACLE SPATIAL REFERENCES 24

7 ARC SDE 25

7.1 ARC SDE JAVA API 25
7.2 ARC SDE DATA MODELING 26
7.3 ARC SDE LOCKING SUPPORT 27
7.4 ARC SDE DATASOURCE 28
7.5 ARC SDE REFERENCES 28

1 INTRODUCTION

This document outlines the differences between a number of data sources used by GeoServer. GeoServer is the reference implementation of the Open GIS Consortium's Web Feature Server Specification.

This document focuses on the differences between PostGIS, ArcSDE and Oracle. Data modeling, Locking and Java connectivity differences are highlighted.

2 WEB FEATURE SERVER REQUIREMENTS

A Web Feature Server (WFS) distributes feature information over the web using XML. The Open GIS Consortium provides the Web Feature Server Implementation Specification. This specification also allows for modification and long transaction support.

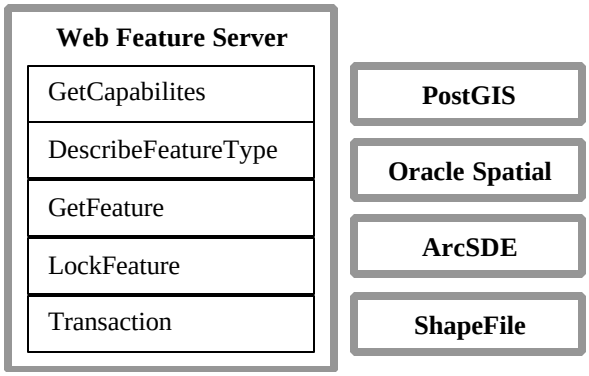

Figure 1: Web Feature Server

Geographic Markup Language (GML) is used to communicate feature information to clients. The Open GIS Consortium also specifies provides the GML Specification.

2.1 Web Feature Server Operations

The Web Feature Server Specification mandates that the following operations be supported:

- **GetCapabilities:**
  Describes the WFS Operations supported and the available Feature Types
- **DescribeFeatureType:**
  Returns an XML Schema document describing the requested FeatureType
- **GetFeature:**
  Returns feature information, from multiple FeatureTypes, that will validate against the DescribeFeatureType supplied XML Schema

The WFS Specification makes use of a Feature ID (or fid) to track features across operations.

2.1.1 Web Feature Server Requirements

Database support:

- Require a representation of feature information
- Require representation of feature schema
- Recommend supporting for fid generation

2.2 Transaction Web Feature Server Operations

In order to support transactions a WFS must support the following optional operation:

- **Transaction**:
  Transaction Operation supports modification of feature information.
  - Multiple Sub-Operations:
    The transaction operation consists of a number of DELETE, INSERT and UPDATE elements in a single request.
  - Partial Results:
    A request may ask a result of both successful modifications and failures.
  - Multiple FeatureTypes:
    Modifications on a number of different FeatureTypes from different data sources.

Web Feature Servers that support this operation are called Transaction Web Feature Servers.

2.2.1 Transaction Web Feature Server Requirements

Database support:

- Requires transaction support

Accounting for partial success without having errors terminate the transaction is particularly difficult.

2.3 Web Feature Server Locking

In order to support long transactions a Web Feature Server provides the following optional locking operations:

- **GetLock:**
  Requests a LockID for a number of features specified by fid or via a filter.
  Interesting properties of this request:
  - Partial Results:
    A lock may be returned that lists both features it was able to lock and features it could not lock
  - Duration:
    Lock requests can be requested for a duration, after which they are no longer valid
  - LockIDs:
    The returned LockID can be used for later Transaction operations.
- **GetFeature:** (extended with **GetFeatureWithLock**)
  The GetFeature operation can be extended to simultaneously perform a locking operation
- **Transaction:** (extended with LockID Element)
  Transaction Operations can now use a number of user supplied LockIDs

2.3.1 Web Feature Server Locking Requirements

Required locking support:

- Require locks on a per feature basis
  (This is often row based locking depending on the data model)
- Require lock persistence
- Require user supplied Authorization Ids for Lock Identification
- Require lock access based on Authorization IDs
- Require locks to support a timeout
- Require compatibility with other (table or row) based locking systems

2.4 Data Modeling

The Web Feature Server is required to Implement the Simple Feature Data Model used by GML 2.1.1.

A Feature:

- is a representation of a real-world geographical object
- Contains geometric information describing location and/or shape
- Contains attributes information describing properties and measurements

The GML 3 specification has been introduced and will probably be a future requirement.

2.5 Geographic Markup Language

The Geographic Markup Language (GML) is used by the Web Feature Server Implementation Specification to describe feature information.

2.5.1 GML Version: 2.1.2

GML Version 2.1.2 provides an XML representation for Simple Features. GML2 Features allow for simple (non-nested) properties and a limited Geometry model.

GML2 Geometry Model:

- geometric properties are represented with two dimensional coordinates
- linear interpolation used to delineate curves

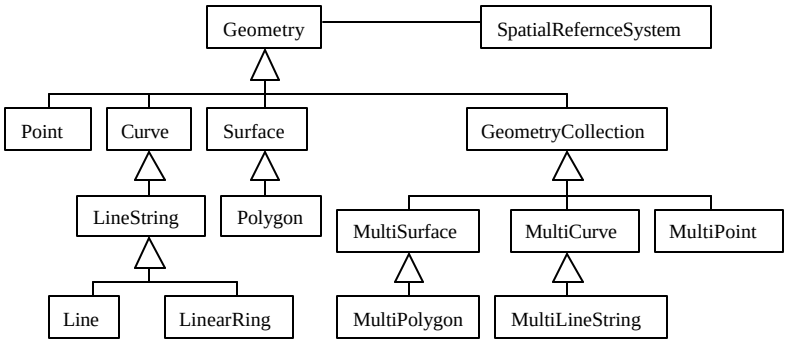

Figure 2: GML2 Geometry

2.5.2 GML Version: 3

Recently the GML specification has been greatly expanded to agree with ISO specifications. GML 3 has expanded beyond providing support for simple features.

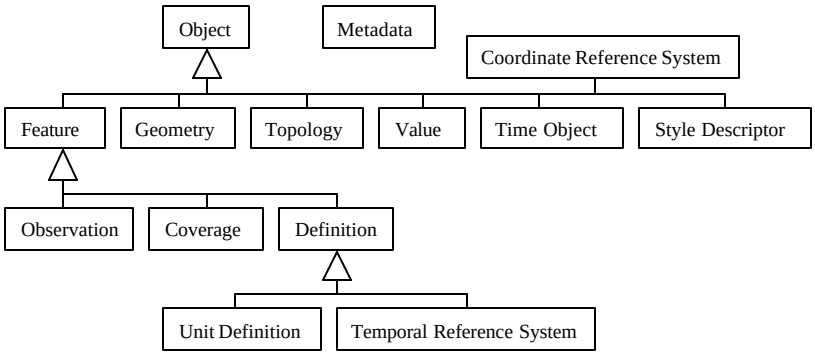

Figure 3: GML3 Data Model

The GML3 Data Model provides:

- Explicit support for complex properties, temporal properties, coverages, and observations
- Spatial and Temporal reference systems
- Style descriptions for rendering.
- Conform to ISO standards on Spatial, Temporal Schema, Encoding and Coverages

The additional complexity included in GML 3 provides for the concept of nested Feature Information. Nested feature information is often required for Object-Oriented Data Models.

The GML3 Data Model for geometric information has similarly been extended.

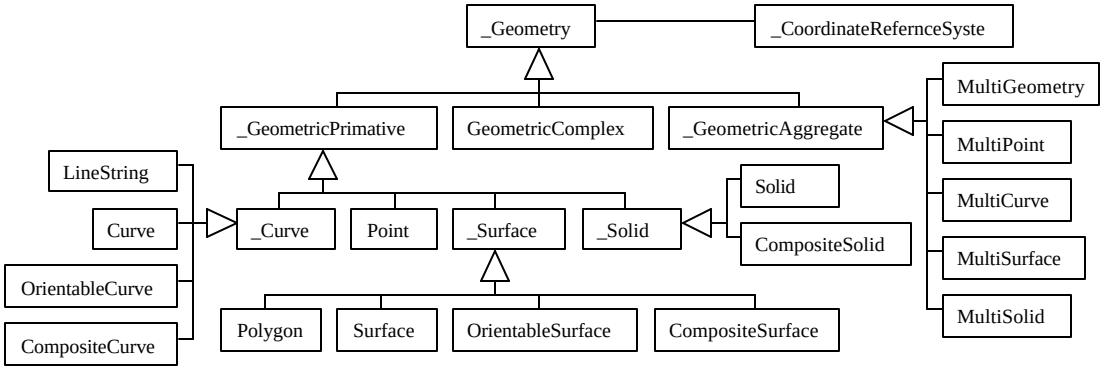

Figure 4: GML3 Geometry

GML3 Geometry Type extends the 2D linear features of GML2 with complex, non-linear, 3D geometry.

GML 3 is eight times the size of GML 2.

2.5.3 Simple Feature Specification for SQL

The Open GIS Consortium's Simple Feature Specification for SQL provides a baseline we can use in evaluating Database Spatial Modeling capabilities.

The Simple Feature Specification for SQL defines a Geometry type, and subclasses, as follows:

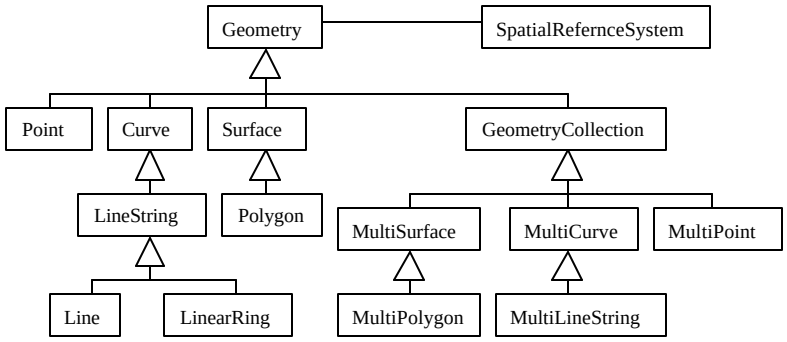

Figure 5: Geometry Type

A number of Geometry methods are required:

Basic Geometry Methods	**Description**
Dimension():int	Dimension (spec is limited to two dimensions)
GeometryType:str	Name of the Geometry subtype
SRID():int	Spatial Reference System ID
Envelope():geom	Bounding Box
AsText():str	Well-Known Text representation
AsBinary():bin	Well-Known Binary representation
IsEmpty():int	1 for empty geometry ∅ for coordinate space
IsSimple():int	1 for no anomalous points (e.g. self intersection)
Boundary():geom	Closure of the combinatorial boundary
Spatial Relationships	**Description**
Equals(geom):int	1 for spatially equal
Disjoint(geom):int	1 for spatially disjoint
Intersects(geom):int	1 for spatially intersects
Touches(geom):int	1 for spatially touches
Crosses(geom):int	1 for spatially crosses
Within(geom):int	1 for spatially wihin
Contains(geom):int	1 for spatially contains
Overlaps(geom):int	1 for spatially overlaps
Relate(geom,str):int	1 for spatial relation described by pattern
Spatial Analysis	**Description**
Distance(geom):double	shortest distance
Buffer(double):geom	geometry within provided distance
ConvexHull():geom	convex hull of Geometry
Intersection(geom):geom	point set intersection between geometries
Union(geom):geom	point set union between geometries
Difference(geom):geom	point set difference between geometries
SymDifference(geom):geom	Point set symmetric difference between geometries

Additional Geometry Type specific operations are defined.

2.5.4 Geometry Representation

The Simple Features Specification for SQL defines two representations for Geometry Information:

- Well-Known Text
- Well-Known Binary

2.5.5 Feature Tables

The Simple Features Specification for SQL defines a schema for storing geometry metadata using SQL92:

- GEOMETRY_COLUMNS:
  Table with a row for each geometry column describing:
  - Describes the feature table and column it is describing
  - Contains metadata: SRID, Geometry Type and Coordinate Dimension
  - Describes the Geometry Table that stores the instances
  - Describes how instances are stored:

```
GEOMETRY_COLUNMNS
  F_TABLE_CATALOG, F_TABLE_SCHEMA, F_TABLE_NAME,F_GEOMETRY_COLUMN
  G_TABLE_CATALOG, G_TABLE_SCHEMA, G_TABLE_NAME
  STORAGE_TYPE
  GEOMETRY_TYPE
  COORD_DIMENSION
  MAX_PPR
  SRID
```

- SPATIAL_REFERENCE_SYSTEM:
  Stores information on each Spatial Reference System used in the database:

```
SPATIAL_REF_SYS
  SRID                    Spatial Reference System ID
  AUTH_NAME               Authority maintaining Spatial Reference System
  AUTH_SRID               Name of Spatial Reference System
  SRTEXT                  Well-Known Text representation
```

- Feature Tables:
  Geometries are stored as foreign keys into a Geometry Table.

- Geometry Tables:
  Geometry information is stored in a table as a Binary Large Object using the Well-Known Binary format:

```
Geometry Table (
  GID                 Geometry ID
  XMIN,YMIN,XMAX,YMAX bounding information
  GEOMETRY            Well-Known Binary Geometry
```

- Geometry Tables:
  Geometry information is stored across a series of rows:

```
Geometry Table (normalized)
  GID                    Geometry ID
  ESEQ                   Geometry Element
  ETYPE                  Geometry Type
  SEQ                    Sequencing for multi row geometries
  X0,Y0,X1,Y1,X2,Y3,X4,Y4 Coordinate information
```

- GEOMETRY Well-Known Binary Geometry

2.6 References

Web Feature Server Implementation Specification,
http://www.opengis.org/techno/specs/02-058.pdf

Open GIS Geography Markup Language (GML) Implementation Specification,
http://www.opengis.org/techno/documents/02-023r4.pdf

Simple Features Specification for SQL,
http://www.opengis.org/techno/specs/99-049.pdf

3 GeoServer Data Access

GeoServer is an open source implementation of the Open GIS Consortium's Web Feature Server Specification. The project's goal is to be the reference implementation of this specification.

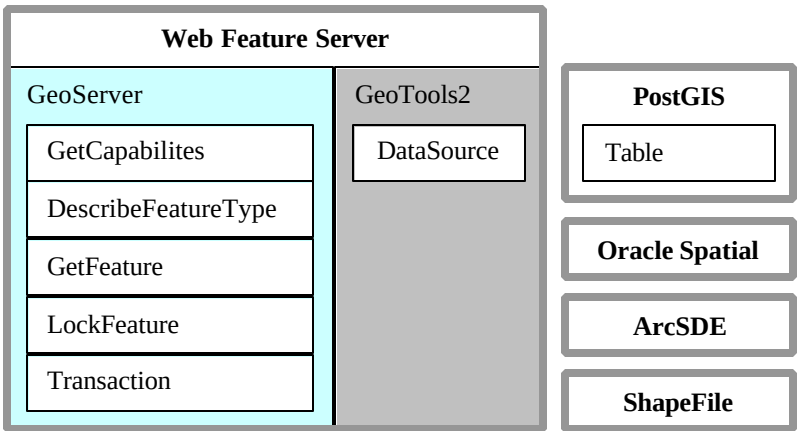

Figure 6: GeoServer Layer Diagram

This diagram shows the main layers in the GeoServer application:

- GeoServer – implements the WFS Specification
- GeoTools2 – provides the DataSource API to access feature information

Currently there are DataSource implementations for:

- Postgis
- Shape files
- OracleSpatial

Work has started on an ArcSDE DataSource

3.1 Geotools2 Data Access

The data package in Geotools2 provides data access.

- DataSource – interface for single Feature Type access
- DataSourceFactorySpi – interface for creating DataSource instances
- DataSourceFactoryFinder – DataSource plug-in

3.1.1 DataSource Interface

The DataSource Interface:

- Provides access to a single database table
- Provides supports for transactions
- Provides support for nested geospatial information

```
public interface DataSource {
   void getFeatures(FeatureCollection collection, Query query);
   void getFeatures(FeatureCollection collection, Filter filter);
   FeatureCollection getFeatures(Query query);
   FeatureCollection getFeatures(Filter filter);
   FeatureCollection getFeatures();
   Set addFeatures(FeatureCollection collection);
   void removeFeatures(Filter filter);
   void modifyFeatures(AttributeType[] type, Object[] value, Filter filter);
   void modifyFeatures(AttributeType type, Object value, Filter filter);
   void setFeatures(FeatureCollection collection);
   void commit();
   void rollback();
   void setAutoCommit(boolean autoCommit);
   boolean getAutoCommit();
   DataSourceMetaData getMetaData();
   FeatureType getSchema();
   void abortLoading();
   Envelope getBounds();
   Envelope getBbox(boolean speed);
}
```

3.1.2 TransactionalDataSource

The DataSource API does not currently support locking.

The proposed TransactionDataSource extension adds locking support:

```
public interface TransactionalDataSource extends DataSource {
   setCurrentLock(Lock)
   lockFeatures();
   lockFeatures(Filter);
   lockFeatures(Query);
   setAuthorization(String[])
   unLockFeatures()
   unLockFeatures(Filter)
   unLockFeatures(Query)
}
```

3.1.3 DataSource Plug-Ins

Geotool2 provides a DataSource plug-in facility:

- DataSource must implement the FeatureFactorSpi interface

```
public interface DataSourceFactorySpi {
   boolean canProcess(Map parms);
   DataSource createDataSource(Map params);
   String getDescription()
}
```

- The jar containing the DataSource must provide a register the DataSource `META-INF/services/org.geotools.data.DataSourceFactorySpi:`

```
org.geotools.data.postgis.PostgisDataSource
```

DataSourceFinder is used to handle DataSource creation:

```
HashMap params = new HashMap();
params.put("dbtype", "postgis");
params.put("host","localhost");
params.put("port", "5432");
params.put("database","test");
params.put("user","test");
params.put("passwd","test");
params.put("table","roads");

DataSource ds = DataSourceFinder.getDataSource(params);
```

3.1.4 JDBC Connection Management

Currently each DataSource maintains it's own JDBC Connection to a database. To mitigate this overhead, connection pooling is planned for geotools2.

3.2 GeoServer Feature Type Access

GeoServer maintains a collection of TypeInfo classes: one for each feature type served by the application.

3.2.1 Feature Type Configuration

GeoServer is configured to serve up information according to Feature Type. The user supplies connection information including a descriptor used to locate the correct DataSource.

Sample Postgis feature type:

```
<?xml version="1.0" encoding="UTF-8" ?>
<featureType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Name>roads</Name>
   <Title>Roads</Title>
   <Abstract>This is an example road dataset</Abstract>
   <Keywords>roads, sample</Keywords>
   <SRS>32118</SRS>
   <LatLonBoundingBox minx="-74.27000" miny="40.50000"
                      maxx="-73.80000" maxy="40.94000" />
   <DatasourceParams>
      <host>localhost</host>
      <port>5432</port>
      <database>test</database>
      <user>test</user>
      <passwd>test</passwd>
      <table>road</table>
      <dbtype>postgis</dbtype>
   </DatasourceParams>
</featureType>
```

The DatasourceParams element above is used to specify the Postgis database connection information.

Sample shape file DatasourceParams element:

```
<DatasourceParams>
   <filename>landmarks.shp</filename>
</DatasourceParams>
```

The DatasourceParams information will be used in conjunction with DataSourceFinder to provide a connection.

3.2.2 TypeInfo DataSource Use

TypeInfo instances maintain two geotools2 DataSource connections:

- FeatureDS - Used for read access
- TransactionDS- Used for transaction and locking operations

4 DATABASE COMPARISON

The following table provides a summary of the capabilities of Postgis, Oracle Spatial and ArcSDE in relation to our Web Feature Server Requirements.

WFS Requirements	**PostGIS**	**Oracle**	**ArcSDE**
Feature storage	✓	✓	✓
Feature Schema	✓	✓	✓
Feature ID Generation			
Transaction WFS Requirements	**PostGIS**	**Oracle**	**ArcSDE**
Transaction Support	✓	✓	✓
Locking Support	**PostGIS**	**Oracle**	**ArcSDE**
Per feature locks			✓
Lock persistence			✓
Lock timeout			
Simple Feature Support	**PostGIS**	**Oracle**	**ArcSDE**
Simple Properties	✓	✓	✓
Multiple Geometries per Feature	✓	✓	✓
Geometry Support	**PostGIS**	**Oracle**	**ArcSDE**
Simple Feature Geometry Model	✓	✓*	✓
Well Known Text representation	✓	✓	✓
Well Known Binary representation	✓	✓	✓
Bounding Box	✓	✓	✓
Spatial Relationships	✓	✓	✓**
Spatial Analysis	✓	✓	✓**
Geotools2	**PostGIS**	**Oracle**	**ArcSDE**
DataSource	✓	✓	
Database Connection	**PostGIS**	**Oracle**	**ArcSDE**
JDBC Support	✓	✓	
JDBC Geometry Data Type Extension	✓	✓	
Custom Connection API			✓

* Oracle does not provide a representation of the Shape Geometry
** ArcSDE Geometric functions may be performed on client side

5 POSTGIS

The reference DataSource implementation used by GeoServer is Postgis. Postgis is an extension of the PostgreSQL Database modified for use with spatial data. The spatial extensions are conformant with the "Simple Features Specification for SQL" published by the Open GIS Consortium.

5.1 PostgreSQL JDBC Drivers

The PostgreSQL JDBC drivers provide support for Postgis access.

- JDBC1
  Supports JDK 1.1, 1.2, 1.3 and 1.4
- JDBC2
  Supports JDK 1.2, 1.3 and 1.4
- JDBC2+
  Supports JDK 1.3 only (uses optional javax.sql.* packages)
- JDBC3
  Supports JDK 1.4

The driver must be included in your `CLASSPATH`. Postgres must be configured to allow JDBC connection. PostgreSQL needs to be started with `-i` in order to accept Internet connections. You will also need to configure your `pg_hba.conf` to allow access.

PostgreSQL uses the following JDBC URL:

```
jdbc:postgresql://host/database
jdbc:postgresql://host:port/database
```

The PostgreSQL driver provides transaction support using the `setAutoCommit` method.

The PostgreSQL driver provides several extensions to the JDBC specification:

- AddDataType extension allows a connection provide custom mappings for Data Types
- Fastpath extension provides access to database functions

By using postgis.jar, and the PostgreSQL JDBC extensions, PostgreSQL data types can be mapped to Java objects.

```
Connection db = Driver.getConnection(url, username, password);
PGConnection myconn = (PGConnection) db;
myconn.addDataType("geometry","org.postgis.PGgeometry");
Statement s = conn.createStatement();
ResultSet r = s.executeQuery("SELECT AsText(geom) AS geom FROM roads WHERE
id=1");
r.next();
PGgeometry geom = (PGgeometry)r.getObject(1);
```

5.2 PostGIS Data Modeling

PostGIS data modeling capabilities provides a mapping for feature information, and geospatial operations.

Open GIS Consortium Standard compliance:

- Postgis is in compliance testing against the Simple Features Specification for SQL.
- Postgis does not support the GML model of nested feature information.

Postgis capabilities:

- Supports the Well-Known Text representation of Spatial Objects including point, line, polygon, multipoint, multiline, multipolygon, and geometrycollections:

```
SELECT AsText(geom) AS OGCGeom FROM landmarks WHERE id=1;
POINT(-126.4 45.32)
```

- Supports the Well-Known Binary representation of Spatial Objects
- Implements the following functions:

```
asBinary( geometry )
dimension( geometry )
isEmpty( geometry )
isSimple( geometry )
boundary( geometry )
equals( geometry )
disjoint( geometry, geometry )
intersects( geometry, geometry )
touches( geometry, geometry )
crosses( geometry, geometry )
within( geometry, geometry )
overlaps( geometry, geometry )
contains( geometry, geometry )
intersects( geometry, geometry )
relate( geometry, geometry, pattern )
buffer( geometry, double )
convexhull( geometry )
intersection( geometry, geometry )
geomunion( geometry, geometry )
difference( geometry, geometry )
envelope( geometry )
geometryType( geometry )
AsText( geometry )
SRID( geometry )
```

- Provides Spatial Referencing System Identifier (SRID) support:

```
GeometryFromText('POINT(-126.4 45.32)', 312 );
```

- Provides metadata tables:

```
Table
SPATIAL_REF_SYS        holds SRID descriptions
GEOMETRY_COLUMNS
```

- PostGIS also provides a "canonical form" which includes SRID number:

```
SELECT geom AS PostGISGeom FROM landmarks WHERE id=1;
SRID=123; POINT(-126.4 45.32)
```

- PostGIS does not contain a Feature Identifier generator, although a combination of an ID column and a generation function can be used.

5.3 PostgreSQL Locking Support

Postgres provides support for table-level locks and Row-level locks. Both these locking mechanisms are for the duration of a transaction, and do not provide persistence. In addition PostgreSQL makes internal use of page-level shared/exclusive locks.

The PostgreSQL locking system will need to be extended to support a locking table in order to meet our locking requirements. Our document on Long Transaction Support details the required changes.

5.3.1 Table-Level Locks

Postgres provides the following table level lock modes:

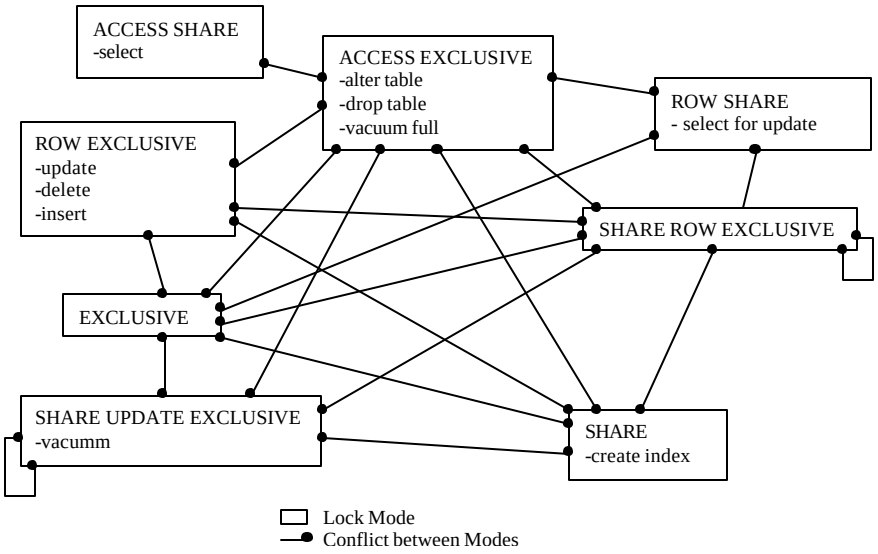

Figure 7: PostgreSQL Table Lock Modes

A transaction can acquire table level locks providing no conflicting locks are held by another transaction. Many PostgreSQL commands implicitly acquire a lock to perform their function.

Table Locking Example:

- A transaction using the SELECT command to read data implicitly requires an ACCESS SHARE lock and can be blocked by a transaction holding an ACCESS EXCLUSIVE lock.
- A transaction can explicitly acquire an ACCESS EXCLUSIVE lock:

```
LOCK TABLE roads IN ACCESS EXCLUSIVE MODE;
```

5.3.2 Row-Level Locks

Postgres also provides support for row-level locking, implicitly these locks are held during an update or delete operation. Row level locks only prevent write operations and are released when the transaction commits or rollbacks.

Using SELECT FOR UPDATE to explicitly acquire a row-level lock:

```
SELCT FROM roads WHERE fid=3 FOR UPDATE;
```

The SELECT FOR UPDATE command implicitly requires a ROW SHARE table lock in order to execute.

5.4 PostGIS DataSource

The geotools2 library uses the PostGIS DataSource as its reference implementation.

The PostGIS DataSource capabilities:

- Support for OCG data types using WKT
- Use of PostGIS bounding box operations
- In-Process FID generation

Nested Features are not supported.

5.5 Postgis References

PostgreSQL,
http://www.postgresql.org/

PostgreSQL JDBC Driver,
http://jdbc.postgresql.org/

PostGIS Manual,
http://postgis.refractions.net/docs/

6 ORACLE SPATIAL

Oracle Spatial is an option for Oracle9i Enterprise Edition that provides spatial features to support GIS applications. Oracle Locator provides limited GIS functionality at a different price point:

6.1 Oracle Spatial JDBC Drivers

The Oracle Spatial JDBC driver and utilities are available for several different Java Runtime Environments.

For Java 1.4:

- ojdbc14.jar – JDBC driver
- ocrs12.zip – additional row set support
- sdoapi.zip – Spatial Data Object API

These files will need to be placed on your CLASPATH.

Oracle Spatial uses the following JDBC URL:

```
jdbc:oracle:thin:@host:port:instance
```

The Oracle Spatial driver provides transaction support using the `setAutoCommit` method.

The Spatial Data Object API provides Geometry Adapter Framework for mapping of spatial data types to other formats. A plug-in System based on the provided GeometryAdapter interface. The framework comes with Adapters for Well-Known Text and Well-Known Binary.

Example use:

```
Connection conn = DriverManager.getConnection(url ,user, password );
SRManager manager = OraSpatialManager.getSpatialReferenceManager(conn);
SpatialReference sr = manager.retrieve(metaData.getSpatialReferenceID());
GeometryFactory gFact = OraSpatialManager.getGeometryFactory(sr);

AdapterSDO adaptersdo = new AdapterSDO( gFact, conn );
AdapterJTS adapterJTS = new AdapterJTS( gFact );

Statement s = conn.createStatement();
ResultSet r = s.executeQuery("SELECT AsText(geom) AS geom FROM roads WHERE
id=1");
r.next();
Object = adapterJTS.exportGeometry(
    com.vividsolutions.jts.geom.Geometry.class,
    adaptersdo.importGeometry( result.getObject(1) )
);
```

6.2 Oracle Spatial Data Modeling

Location data is modeled in layers that share a common coordinate system. A layer can represent information ranging from a table to a complete database.

Oracle Spatial supports two representations of geometric data:

- A table based object-relational representation in which each row can maintain a geometry instance in a column of type MDSYS.SDO_GEOMETRY

```
CREATE TABLE roads
    (id NUMBER PRIMARY KEY, name VARCHAR2(32), geom MDSYS.SDO_GEOMETRY);
INSERT INTO roads VALUES(1, 'road one',
        MDSYS.SDO_GEOMETRY( 2003, NULL, NULL,
                            MDSYS_ELEM_INFO_ARRAY(1,2002,2),
                            MDSYS.SDO_ORDINATE_ARRAY( 1,1, 5,7)
        )
);
```

- A table based relational model representation in which one or more rows with a prescribed set of columns of type NUMBER are used.

Linear Referencing of measurements along feature information is supported.

Available Spatial Data Types:

```
Points, Lines, Polygons, Complex Polygons With Holes, Arc Strings, Compound
Polygons, Circles, Rectangles
```

The Spatial Data types are similar to the OCG Simple Feature Model. Mappings exists for all OCG data types except Surface. The OCG data types can map all Oracle Spatial types with the exception of Line Curves and Polygon Curves.

Spatial Operations:

```
SDO_GEOM.RELATE( geom1, dim1, mask, geom2, dim2 )
Where mask is: ANYINTERACT, CONTAINS, COVEREDBY, COVERS, DISJOINT,
               EQUAL, INSIDE, OVERLAPBDYDISJOINT, OVERLAPBDYINTERSECT, TOUCH
SDO_GEOM.SDO_AREA( geom, dim, tolerance )
SDO_GEOM.SDO_BUFFER( geom, dim, distance )
SDO_GEOM.SDO_CENTROID( geom, dim )
SDO_GEOM.SDO_CONVEXHULL( geom, dim )
SDO_GEOM.SDO_DIFFERENCE( geom1, dim1, geom2, dim2 )
SDO_GEOM.SDO_DISTANCE( geom1, dim1, geom2, dim2 )
SDO_GEOM.SDO_INTERSECTION( geom1, dim1, geom2, dim2 );
SDO_GEOM.SDO_LENGTH( geom, dim )
SDO_GEOM.SDO_POINTONSURFACE( geom, dim )
SDO_GEOM.SDO_UNION( geom1, dim1, geom2, dim2 );
SDO_GEOM.SDO_XOR( geom1, dim1, geom2, dim2 );
SDO_GEOM.VALIDATE_GEOMETRY( geom, dim )
SDO_GEOM.VALIDATE_LAYER( geom_table, geom_column, pkey_column, result_table)
SDO_GEOM.WITHIN_DISTANCE( geom1, dim1, distance, geom2, dim2 );
```

- Spatial Aggregate functions for determining the union, centroid, convex hull and minimum bounding rectangle:

```
SELECT SDO_AGGR_MBR( roads.geom ) FROM roads;
```

6.3 Oracle Spatial Locking Support

Oracle provides the following locking support:

- DML locks used to protect data at the row and table level
- Dictionary locks used protect database structure

6.3.1 Table Locking

Table locking modes:

- EXCLUSIVE (X) - Allows queries but nothing else.
- SHARE (S) - Allows queries but not updates.
- ROW SHARE (RS) - Allows to table but not exclusive locks.
- ROW EXCLUSIVE (RX)- Allows access to table but not exclusive or share mode locks. Updates, deletes and inserts use this lock mode.
- SHARE ROW EXCLUSIVE (SRX) – Allows row access, prevents share locks and updates.

These locks are held for the duration of a transaction.

Table Lock Example:

```
LOCK TABLE roads IN SHARE MODE NOWAIT;
```

6.3.2 Row Locking

Row locking is provided using the SELECT FOR UPDATE command in a manner similar to Postgis.

Row Lock Example:

```
SELECT * FROM roads WHERE id=1 FOR UPDATE OF geom NOWAIT;
```

The OF keyword allows the identification of columns to be modified.

6.3.3 Locking Evaluation

Oracle Spatial does not meet our locking requirements (authorization IDs, timeouts, etc…). By using Oracle's trigger mechanism we can produce a locking table in a similar manner to our Postgis implementation.

Oracle does provide long transaction support via row versioning. Long transaction support is built around the concept of workspaces. Differences introduced by operations on the workspace are maintained at the row level. Workspaces can be branched by creating savepoints, allowing the spawning of additional workspaces or a point in time to rollback to. The merging of workspaces results in a conflict view that needs to be manually resolved.

6.4 Oracle Spatial DataSource

Sean Geogheqan has recently added an Oracle Spatial data source to the geotools2 development library.

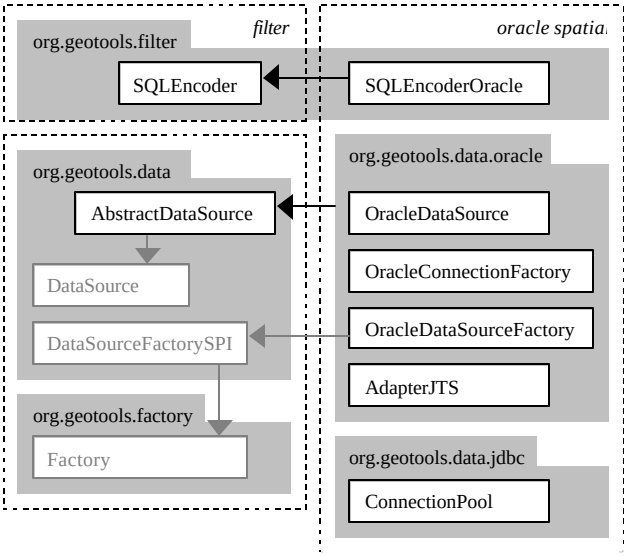

Figure 8: OracleDataSource

Oracle Spatial Geotools2 module:

- OracleDataSource and OracleDataSourceFactory provide Oracle Spatial access for Geotools2.
- ConnectionPool and OracleConnectionFactory mitigate the number of database connections required.
- SQLEncoderOracle provides the SQL Statement construction. It also provides Oracle Spatial Bounding Box filter support.
- AdapterJTS is a plug-in to the Spatial Data Objects Geometry Adapter Framework allowing the conversion to the Java Topology Toolkit representation used by geotools2.

6.5 Oracle Spatial References

Oracle Info,
http://www.ilook.fsnet.co.uk/ora_sql/sql_03.htm

Managing Long Transaction Using Standard DBMS Technology,
http://www.ilook.fsnet.co.uk/ora_sql/sql_03.htm

Oracle Spatial Java Library User's Guide,
http://otn.oracle.com/docs/products/spatial/pdf/sdoapi.pdf

Oracle Spatial User's Guide and Reference,
http://homepages.feis.herts.ac.uk/~oradoc/inter.816/a77132/toc.htm

7 ARC SDE

ArcSDE serves spatial information to an ArcView, ArcEdit and ArcInfo.

ArcSDE provides database access for IBM DB2, IBM Informix, Microsoft SQL Server and Oracle. ArcSDE for Coverages provides access to data stored in ESRI file formats.

7.1 Arc SDE Java API

ESRI provides an ArcSDE Java API in two parts:

- Client Package:
  Provides connection support and Feature access.
- Geometry Package:
  Uses JNI to access native code for spatial operations and relations.

The ArcSDE Java API is distributed as a series of JAR files and native libraries:

- jsde83_sdk.jar
  Contains client and geometry packages
- jsg83.dll / libjsg83.so / libjsg83.sl
  Platform specific native code for geometry package
- sdejavautil.dll / libsdejavautil.so / libsdejavautil.so
  Platform specific native code for SeInstance start function

7.1.1 Using ArcSDE for Data Access

The ArcSDE Java API client package provides a connection mechanism analogous to JDBC.

```
SeConnection conn=new SeConnection(server, instance, database,user, password);
SeLayer layer = new SeLayer( conn, “roads”, “geom” );
SeSqlConstruct sqlConstruct = new SeSqlConstruct( layer.getName() );
String[] cols = new String[2]{“name”,layer.getSpatialColumn()};
SeQuery query = new SeQuery( conn, cols, sqlConstruct );

query.prepareQuery();
query.execute();

SeRow row = query.fetch();
String name = Row.getString(0);
Shape geom = Row.getShape(1);
query.close();
```

Geometry is represented using the OCG Well-Known Text and Well-Known Binary standards. ArcSDE compressed binary representation and ESRI Shape Representation can also be used.

Point Geometry represented using Well-Known Text:

```
pointfromtext(‘point zm(10.98 29.91 10.2 9.1)’, 1 )
```

The Well-Known Text standard has been extended by ArcSDE to include Elevations and Measures.

7.1.2 Using ArcSDE Java API Geometry Package with JDBC

The ArcSDE Java API supports limited direct integration with JDBC. The geometry package can be used for its definitions of spatial information.

Integration with JDBC is accomplished by using the geometry package to extract spatial information form BLOBS using Well-Known Binary. There is no support for spatial queries using this technique.

7.2 Arc SDE Data Modeling

ArcSDE provides a consistent RDBMS mapping of Spatial Information across a several database implementations. The different ArcSDE backends provide mappings to native database geometry storage:

Database	Storage
IBM DB2	Extension of SQL Type System
Informix	Extension of SQL Type System
Microsodt SQL Server	Blob
Oracle	Blob
	Extension of SQL Type System
	Normalized Schema

ArcSDE provides a Geometry Metadata service as a series of support tables:

- LAYERS:
  ArcSDE database specific geometry description, a row is maintained for each spatial column in the database.
- SPATIAL_REF_SYS:
  Required by OCG SQL specification, extended with for ArcSDE coordinate transformations.
- GEOMETRY_COLUMNS:
  Defined for compatibility with the OpenGIS SQL specification

Raster Modeling is supported using a series of raster support tables:

- RASTER_COLUMNS:
  System table used to store a mapping from the user's raster table columns to the Raster Table responsible for storage.
  This table is indexed by RasterColumnID.
- SDE_RAS_*RasterColumnID*:
  Raster table responsible for storage of raster information.
- SDE_BND_ *RasterColumnID*:
  Raster Band Table provides a mapping for each band of raster information.
- SDE_BLK_ *RasterColumnID*:
  Stores actual pixels of the raster image. Pixels are divided into tiles.
- SDE_AUX_ *RasterColumnID*:
  Stores metadata, such as Color maps, and statistics.

7.3 Arc SDE Locking Support

7.3.1 Application Objectlocks

ArcSDE provides an ObjectLock API to allow application defined locking schemes. The ArcSDE C API supports:

```
Lock Types:
SE_OBJECTLOCK_SHARED_LOCK
SE_OBJECTLOCK_EXCLUSIVE_LOCK
SE_OBJECTLOCK_NO_LOCK_SET
Definition:
SE_objectlockinfo_get_lock_mode
SE_objectlockinfo_get_object_id
SE_objectlockinfo_get_object_type
SE_objectlockinfo_get_onwer
SE_objectlockinfo_set_object_id
SE_objectlockinfo_set_object_type
Locking:
SE_objectlock_get_list
SE_objectlock_lock
```

The current Java implementation is limited to ESRI internal application development and only supports the locking of registered tables. Objectlocks are not persistent and released when an application disconnects from ArcSDE.

7.3.2 Row Lock Support

Arc SDE provides persistent row-locks that are held per user.

Row- Locks are obtained using the SE_table_* functions:

```
SETable.addRowLocks()
SETable.getRowLocks()
SETable.getRowLocksByUser()
SETable.removeRowLocks()
```

These functions work with a list of row_id.

7.3.3 Implementation Details

ArcSDE takes a different approach to locking depending on the native support provided by the underlying database.

Database	ArcSDE Support
Oracle 8	In Memory Locking
Oracle 8I	System Tables: • LAYERS_LOCKS • OBJECT_LOCKS • STATES_LOCKS • TABLE_LOCKS Stored Procedures: • DBMS_PIPE package store and transmit ArcSDE rowids • DBMS_LOCKS manage session's PROCESS_INFORMATION table

7.4 Arc SDE DataSource

The geotools2 library does not currently support ArcSDE.

ArcSDE Design recommendations:

- Use the ArcSDE API exclusively, rather than the JDBC hybrid support
- Use Well-Known Text to construct JTS Geometries rather than use Arc Geometry classes
- Use Row-Level Locking by associating each GeoServer LockID with a different ArcSDE user

7.5 Arc SDE References

ArcSDE Developer Help 8.3,
http://arcsdeonline.esri.com/index.htm

ArcSDE Configuration and Tuning Guide for Oracle
http://gis.ccom.unh.edu/gis/config_tuning_guide_oracle.pdf

ESRI Software and Open GIS Issues
http://www.esrisa.com.sg/files/pages/pdf%20files/opengispos.pdf