Design Implications

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

1 INTRODUCTION

This document outlines our experience in extending Geotools2 to support locking. The primary focus in on the resulting design changes with respect to our Strong Transaction Support Document.

The focus of our design changes have been:

- To Increase Clarity
- To further align the Locking Support with the Web Feature Server specification.
- To explicitly define expected Locking schematics

We have been pleased with the stability our Design. There have been no significant modifications throughout the implementation process.

2 DESIGN LESSONS

2.1 Successes

The Design for Strong Transaction Support has succeeded in:

- Meeting the Web Feature Server Implementation Specification requirements for locking support.
- Garnering favorable reviews of row-locking and feature-locking support
- Stable rapid evolution through confidence in our testing procedures.

We are looking forward to further development on this project.

2.2 Limitations

Over the course of implementing locking support we encountered several limitations to our original design

2.2.1 Locking Semantics and Naming

The semantics of our initial lock design were inexact and the language confusing. The extract behavior on lock generation and lock conflict was not defined.

The over use of the word “lock” led to confusion between our requirements and industry standard table-lock, row-lock and long term transaction support. The concept of Lock ID was similarly overused.

2.2.2 Transaction LockID with Multiple Authorizations

The Web Feature Server Specification allows for a single Lock ID element per Transaction Request. This is a poor fit with multiple locks required to keep pace with the multi-data source nature of GeoServer.

We had misunderstood the Web Features Server Specification and expected support for multiple Lock IDs. We had planned to use separate Lock ID to represent the locks required for each required Datasource.

2.2.3 Transaction Release MODE SOME Lock Refreshing

The Web Feature Server Transaction Operation supports two lock release modes:

1. ALL
    - Releases all features locked by Lock ID even if the Transaction did not modify them
2. SOME
    - Lock release is limited to features modified by the Transaction
    - Features not modified by the transaction need to have their locks reset to there initial duration

This requirement of the Web Feature Server Implementation Specification was missed. The requirement is located in the Transaction documentation and was not mentioned the Locking State Diagram we used for planning.

2.2.4 GeoServer Testing

Additional testing with the GeoServer would be beneficial.

2.3 Response and Recommendations

We have composed a small collection of recommendations to address the limitations of our original design.

2.3.1 Locking Semantics and Naming

We have decided to call Web Feature Server style locking a feature-lock.

A feature-lock is:

- Defined as locking a Feature ID with an Authorization ID for a Duration
- The requesting of a feature-lock does not block and returns the number of features locked.
- Lock conflict does not block and will result in an error
- Lack of authorization does not block and will result in an error

This definition clearly defines a feature-lock and it capabilities. The Locking API has been clarified and the Postgis tables recast in terms of this new vocabulary.

2.3.2 Transaction LockID with Multiple Authorizations

The contents of the LockID string are implementation defined and may be used to represent several Authorizations.

We can enlist this capability to encode our required Authorization IDs into a single Lock ID. Using a comma to separate out Authorization IDs would be sufficient.

This practice would diverge from the simple examples in the Validating Web Feature Server Specification.

The ability of our Design to request locks from several DataSources at once with a common Authorization ID should mitigate the need for Multiple Authorizations acquired separately. As such it may be the intent of the specification.

2.3.3 Transaction Release MODE SOME Lock Refreshing

The Locking API will need additional methods to capture the new requirement.

The Postgis Locking Table implementation will require and extra column to maintain duration information.

2.3.4 GeoServer Testing

We decided to focus on developing robust test suite. This facilitated our stable evolution of the Locking API.

Additional GeoServer testing will be required.

3 CHANGES TO POSTGIS FEATURE-LOCKING

PostGIS locking facilities have been changed to have a direct mapping to feature-locks. We will now explicitly lock features rather than rows, functions and tables are expressed in terms of feature-locks, FID and Authorization ID.

3.1 PostGIS Feature-Locking

A unique alphanumeric Feature ID (FID) identifies features. In Postgis FID is often generated by combining the Table name and per row GID generated by the shape2pgsql application.

- Changed to a single locking table referenced by table/FID rather than a locking table per FeatureType table referenced by a unique integer key:

```
CREATE TABLE authorization_table (tname text, fid text, expires date,
integer duration, authid text, PRIMARY KEY (tname,fid) );
```

- Switched from explicitly binding trigger functions to an attacher function that does the work and remembers the FID column by FeatureType for you:

```
select AuthroizationTriggerAttacher('geo_table', 'id');
```

- Updating a row has changed to use the word 'Authorization' instead of 'Lock' to avoid confusion:

```
BEGIN;
    select authorizationlock('geo_table','2','2004-01-01','XX22XX');
    select haveAuthorization('XX22XX');
    update geo_table set name = 'Jody' where fid = 'geo_table.2';
END;
```

An example is listed below:

```
CREATE TABLE geo_table (id int primary key,the_geom geometry, name text);
select AuthroizationTriggerAttacher('geo_table', 'id');

insert into geo_table values (1, 'POINT(0 0)', 'dave');
insert into geo_table values (2, 'POINT(1 0)', 'jody');
insert into geo_table values (3, 'POINT(2 0)', 'brent');
insert into geo_table values (4, 'POINT(3 0)', 'justin');

-- this will update one row
BEGIN;
  select authorizationlock('geo_table','2','2004-01-01','XX22XX');
  select haveAuthorization('XX22XX');
  update geo_table set name = 'Jody' where id = 2;
END;
```

4 GEOTOOLS2 LOCKING API

The Locking API has been recast in terms of feature-locks, FIDS and Authorization ID. The Locking API has maintained row-lock support.

4.1.1 LockingDataSource

TransactionalDataSource has changed to LockingDataSource to agree with current usage.

The following extension to DataSource will be required to support locking:

```
class LockingDataSource extends DataSource {
     int    lockFeatures()
     int    lockFeatures(org.geotools.filter.Filter filter)
     int    lockFeatures(org.geotools.data.Query query)
     void   refreshLock(java.lang.String authID)
     void   releaseLock(java.lang.String authID)
     void   setAuthorization(java.util.Set authIds)
     void   setAuthorization(java.lang.String authID)
     void   setFeatureLock(FeatureLock lock)
     void   unLockFeatures()
     void   unLockFeatures(org.geotools.filter.Filter filter)
     void   unLockFeatures(org.geotools.data.Query query)
}
```

All lockFeatureMethods are non-blocking and report the number of successfully locked features. Lock management methods refreshLocks and releaseLocks have been added.

4.1.2 FeatureLock

The class Lock has changed to FeatureLock. The functionality remains the same.

```
class FeatureLock {
     static  FeatureLock generate(long duration)
     static  FeatureLock generate(String name, long duration)
     String              getAuthorization()
     long                getDuration()
}
```

A FeatureLock is expressed in terms of duration to allow for lock refresh.