VersioningDataStore API

This RnD page is set up to explore the pandora's box opened by the postgis-versioned module; and see it it can be cleaned up as a more general purpose solution suitable for use by geogit, arcsde, oracle and postgis-versioned.
Table of contents:

Children:

Reference:

- FeatureVersioning interface api (withdrawn proposal)

Scope

This work is being undertaken by LISAsoft to define a datastore used to store and access revisions. We have not chosen a backend at this time - the following are under consideration:

- postgis-versioned - requires porting to jdbc-ng
- geogit - currently blocked behind GPL license

Also for consideration:

- WFS 2.0 - covers the concept of versioning
- arcsde - gabriel needs to be happy that the interfaces will work to advertise arcsde version / history
- oracle - jody needs to be happy that the interface will work to advertise oracle long term transaction support

Timeline:

- Sept:
  - Prototype (as an unsupported module)
  - JDBC-NG change requests to allow extension (assuming postgis used)
- Oct: QA and Documentation
  - Documentation and Unity Testing 40% resulting in Supported Module
  - Proposal for cross DataStore interfaces
- Nov: Delivery and user acceptance testing

__Version Model__

The use of the word "version" changes between different back-ends - as such I will try and use the following in a consistent manner:

- *__Branch__* - used to represent a complete standalone branch of a dataset; it could be published out as a single shapefile for use offline.
- *__Revision__* - represents a change to a feature; you can review the history of a feature; pulling back each revision from initial creation through to the latest

PostGIS-Versioned

The module would need work to migrate to JDBC-NG (and discussion with Justin as JDBC-NG classes are all marked **final**.

The details on migration to JDBC-NG are slightly off topic (although interesting) - as such a couple of options are listed below:

Option: Extend JDBCDataStore

- VersionedPostGISDataStore extends JDBCDataSTore: (this would violate one of the main advantages of the design; the closed nature forces each implementor to talk to Justin and has resulted in additional implementations being donated to the library)

Option: Modify JDBCDataStore - boil the concept of versioning into the JDBCDataStore; an ability that would not be leveraged by the default implementations. For this to work:

- This would prevent the addition of new methods to the "DataStore" api; it could only be handled at the VersioningFeatureSource, VersioningFeatureStore level
- Extra Strategy: Provide a second strategy object in addition to SQL Dialect; that would allow the SQLDialect to be "Version Aware" when generating select statements; would also need the ability to generate a VersioningFeatureStore as required to support additional methods
- Wrapper Strategy: Provide an SQLDialect "wrapper"

Option: "Pure" DataStoreWrapper

- new VersioningDataStore(PostGISDataStore datastore)
- JG: This direction has the most support because it is a cool idea; however the testing burden here is punishing; I would rather wrap the SQL Dialect used "internally" by the JDBCDataStore and not leave developers with two objects representing the same content.

__Version Model__

This datastore captures:

- changesets; each identified by a revision number
- revision: for each feature

GeoGIT

The GeoGIT project is GPL; in part to make use of Binary XMl from the gvSig project.

Option: Ignore the project as not compatible with GeoTools / LGPL

Option: Ask for the project to be LGPL; and rewrite the binary XML part. Conversation with Gabriel indicates this is around two months to do

Option: Ask for GeoGIT to to be LGPL: and ask gvSIG to release the binary XML part under some license we can use

We understand this project is only taking shape now; and follows a model where a dataset can be "unpacked" into a normal DataStore. Even with this model we would still need the ability to query revision history as per the VersioningDataStore API defined by postgis-versioning.

__Version Model__

- branch
- revision

WFS 2

Justin is implementing WFS 2.0; as such there is a couple things we need to keep in mind.

- ResourceID: provides a model for refering to LATEST and so on. Will probably be a join change proposal with jdeolive

__Query__

Q: Supports the concept of Version; not sure if that is the ArcSDE model (ie a branch of the dataset); or trying to indicate a revision of a feature.

Q: Not sure how to query revision history (ie single feature; get the historical changes)

ArcSDE

TBD

__Version Model__

- Version: captures a version of a dataset; in programmer speak this would be a "branch"
- History: captures changes to individual features; in programmer speak this would be a revision of a feature

Oracle

Oracle supports "long term transaction support"; a model similar to the ArcSDE concept of "versions". Update: They renamed the functionality as "workspace manager".

__Version Model__

For "versioned enabled tables":

- workspace: provides a "branch" offering a complete dataset
- "revision" does not seem to be supplied

__Reference__

- http://www.oracle-base.com/articles/9i/WorkspaceManagement9i.php
- http://www.idevelopment.info/data/Oracle/DBA_tips/Workspace_Manager/WM_1.shtml

Background

Prior work from postgis-versioned module:

- VersioningDataStore.java
- VersioningFeatureStore.java
- VersioningFeatureSource.java