Improve CRSAuthority Concurrency Caching and Connection Use

Motivation:	Allow concurrent access to a CRS authority with caching support
Contact:	Jody Garnett Cory Horner
Tracker:	http://jira.codehaus.org/browse/GEOT-1286
Tagline:	Got CRS?

This page represents the **current** plan; for discussion please check the tracker link above.

Description

Currently, GeoTools is tuned for use by a very small number of concurrent users. When deploying this subsystem in a highly threaded environment with thousands of users some shortcomings are revealed:

- **Allowing Multiple Users** - currently multiple threads are only supported by virtue of being serviced by a cache hit, only one thread is allowed to work on a cache hit at a time.
- **Cache Handling** - we need to ensure the two caching techniques used (`pool` and `findPool`) are able to function in the face of multiple threads (both reading and writing).
- **Connection Issues** - this is a related issue only of interest to `CRSAuthority` implementation making use of an EPSG Database, we need to ensure that supporting multiple threads does not result in Connections being leaked from the `java.sql.DataSource` provided.

Status

This proposal has been accepted; implemented and is waiting the attentions of the Module Maintainer for referencing. We are currently running **tw o** implementations of the various base classes.
This proposal has been approved!

- Andrea Aime +1
- Ian Turton +1
- Justin Deoliveira +1
- Jody Garnett +1
- Martin Desruisseaux +1
- Simone Giannecchini +0

Initial feedback has resulted in the request to combine the "Concurrency and Cache" issue with the "DataSource and Connection" issue as both must be considered together for an effective solution.

Martin (the module maintainer) has also given us permission to rename classes as part of this refactoring - this is GREATLY appreciated as as talking about the difference between `FactoryOnOracleSQL` and `FactoryUsingSQL` has cost a wee bit of sanity around here.

Getting the EPSG authorities in ship shape has been harder than expected; the AuthorityCodes implementation of set was also caught holding onto connections and prepared statements.

Martin also pointed us in the direction of a JSR about caching; we have changed the method signature of ObjectCache in order to be method compatible.

dynamictasklist: task list macros declared inside wiki-markup macros are not supported

Resources

For background reading on the design of the GeoTools referencing system:

- 0 Referencing Overview
- 3 Authorities backed by the EPSG Database
- CRS Authority Allowing Multiple Users
- CRS Authority Cache Handling
- CRS Authority Connection Use
- CRS Authorty Alternative Proposals

Tasks

	no progress		done		impeded		lack mandate/funds/time		volunteer needed

2.4-M4:

1. Jody Garnett Rename CRSAuthortyFactory classes for clarity
2. Cory Horner Isolate ObjectCache
3. Martin Desruisseaux Review ObjectCache API
4. Jody Garnett Review and document ObjectCache
5. Cory Horner Create DefaultObject Cache and stress test
6. Jody Garnett Break BufferedAuthorityFactory into three parts; one for each responsibility

2.4-RC0:

1. Jody Garnett Create WeakObject and FixedObjectCache for stress testing
2. Cory Horner AbstractCachedAuthorityFactory - inject ObjectCache into worker
3. Cory Horner AbstractCachedAuthorityMediator - provide Hints to control Object Pool
4. Jody Garnett Set up OracleDialectEpsgFactory lifecycle for Connection use
5. Jody Garnett Using HsqlDialectEpsgMediator - confirm ObjectPool can manage a single worker
6. Jody Garnett Finish renaming the EpsgFactory classes for clarity
7. Martin Desruisseaux Review and update Naming in response to Implementation
8. Cory Horner Provided a test to Break ObjectCache memory limit
9. Cory Horner Use WeakReferences to keep withing memory limit

2.4-RC1:

1. Martin Desruisseaux Review HSQL implementation and deprecate old approach
2. Jody Garnett Complete documentation
3. Jody Garnett Stress test epsg-oracle module
4. Jody Garnett Bring oracle-epsg up to supported status |

API Changes

Public API change

This results in **no change** to client code - all client code should be making use of one of `AuthorityFactory` sub-interfaces or the CRS facade as documented here:

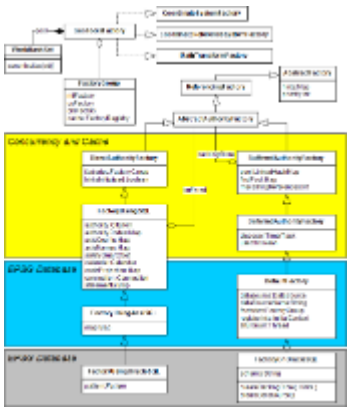
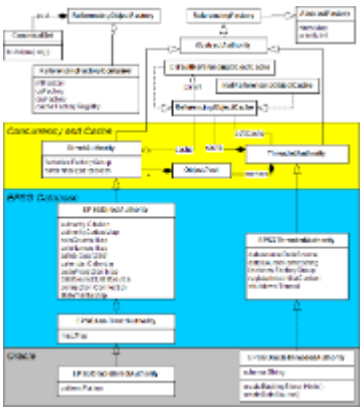
- 01 CRS Helper Class

Internal API change

So what is this change about then? This change request is for the internals of the **referencing** module, and the relationship between the super classes defined therein and the plug in modules providing implementations.

Requests:

- Stop calling everything Factory
- Try and communicate the difference between an authority "using oracle to host the EPSG database" and "using the Oracle SRID wkt definition"
- Keep in mind that most authority factories are about `CoordinateReferenceSystem`, `CoordinateSystem`, `Datum`, `CoordinateOperation`, *etc.*
- Sort order of the names matter for people reading javadocs - they should be able to see alternatives sorted together
  - (note from Martin: I agree about grouping the related alternatives, but I don't think that we should do that through alphabetical order if it produces confusing names. Grouping is Javadoc's job. Current javadoc tool can do that at the packages level. Future javadoc tool will be able to do that at the classes and methods level. So lets not produce bad names for working around a temporary javadoc limitation)
- Please don't hold up your vote over naming - if you see something you want edit the page and vote +1

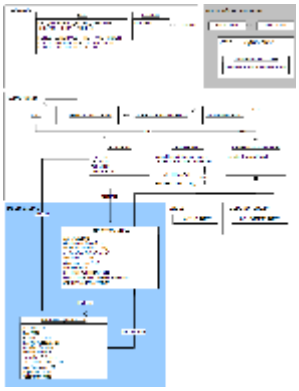
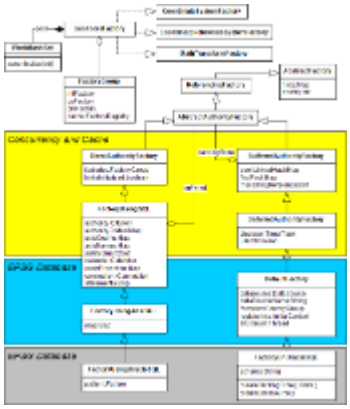
BEFORE	Proposed	AFTER	Role and Responsibility
			Class diagrams, additional diagrams follow.
`AbstractFactory`	unchanged	unchanged	
`ReferencingFactory`	unchanged	unchanged	
`AbstractAuthorityFactory`	`AbstractAuthority`	`AbstractAuthorityFactory`	Authority providing the definition of referencing objects for a code
BEFORE	**Proposed**	**AFTER**	**Role and Responsibility**
`BufferedAuthorityFactory`	`ThreadedAuthority`	`AbstractAuthorityMediator`	Manages a shared cache and worker creation/dispatch/reus for multiple threads
		`CachedAuthorityDecorator`	Decorator used to wrap cache support around an existing authority
		`AbstractCachedAuthorityFactory`	Acts as a super class for authority factories making use of a cache
	`DefaultReferencingObjectCache`	`DefaultReferencingObjectCache`	Shared cache implementation
	`ObjectPool`	`ObjectPool`	An off the shelf component from commons-pool
`DeferredAuthorityFactory`			remove
`DefaultFactory`	`EPSGThreadedAuthority`	`EpsgMediator`	Implementation backed by official EPSG database
`FactoryOnOracleSQL`	`EPSGOracleThreadedAuthority`	`OracleEpsgMediator`	When the EPSG database is loaded int Oracle
BEFORE	**Proposed**	**AFTER**	**Role and Responsibility**
`DirectAuthorityFactory`	`DirectAuthority`	`AbstractCachedAuthorityFactory`	Direct access to an authority
	`NullReferencingObjectCache`	`NullObjectCache`	NullObject used to maintain stand-alone functionality

FactoryUsingSQL	EPSGDirectAuthority	EpsgFactory	Consult an offical EPSG database for CRS definition
		AccessDialectEpsgFactory	Dialect for Access SQ
FactoryUsingAnsiSQL	EPSGAnsiDirectAuthority	AnsiDialectEpsgFactory	Dialect for ANSI SQL
FactoryUsingOracleSQL	EPSGOracleDirectAuthority	OracleDialectEpsgFactory	Dialect for Oracle SQL
BEFORE	**Proposed**	**AFTER**	**Role and Responsibility**
FactoryGroup	ReferencingFactoryContainer	ReferencingFactoryContainer	Holds all the factories needed to create stuff.
GeotoolsFactory	ReferencingObjectFactory	ReferencingObjectFactory	Creates the default implementations provided by the GeoTool library
WeakHashSet	WeakHashSet	WeakHashSet	Uses weak references to store set contents
	CanonicalSet	CanonicalSet	Isolate toUnique method in order to implement "intern" functionality

Design Changes

We are documenting this as a refactoring with BEFORE and AFTER pictures. For design alternatives please review the comments of GEOT-1286

BEFORE

Runtime Overview	Class Diagram	Sequence Diagram
		

For background reading on the design of the GeoTools referencing system:

- 0 Referencing Overview
- 3 Authorities backed by the EPSG Database

Allowing Multiple Users

`FactoryOnOracle` (ie a `BufferedAuthorityFactory`) allows multiple threads, making use of a an internal **pool** as a cache for objects already constructed. In the event of a cache miss the backingStore is used to create the required object.

`FactoryUsingOracleSQL` (ie a `DirectAuthorityFactory` acting as a "backingStore") has **synchronized** each and every public method call (internally it makes use of a Thread lock check to ensure that subclasses do not confuse matters).

When creating compound objects will make a recursive call to its parent **buffered** `FactoryOnOracle`. This recursive relationship is captured in the sequence diagram above.

A Timer is used to dispose of the backingStore when no longer in use.

Cache Handling

`FactoryOnOracle` (ie a `BufferedAuthorityFactory`) makes use a **pool** (a `HashMap` of strong and weak references) in order to track referencing objects created for use by client code. By default, the 20 most recently used objects are hold by strong references, and the remaining ones are hold by weak references. A second cache, **findPool**, makes use of a `HashMap` of `WeakReferences` in order to keep temporary referencing objects created during the use of the find method.

The garbage collector is used to clean out weak references as needed.

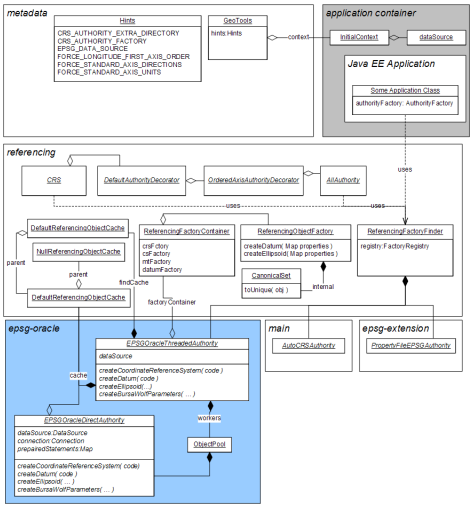
Connection Issues

A single **connection** is opened by `FactoryOnOracle`, and handed over to the backingStore (ie `FactoryUsingSQL`) on construction. This connection is closed after a 20 mins idle perriod (at which point the entire backingStore is shut down). This work is performed by a timer task in `DeferredAuthorityFactory`, not to be confused with the thread **shutdown** in `DefaultFactory`, which is a shutdown hook used to ensure the connection is closed at JVM shutdown time.

AFTER

The referencing module functions as normal, classes have been renamed according to function:

Runtime Overview



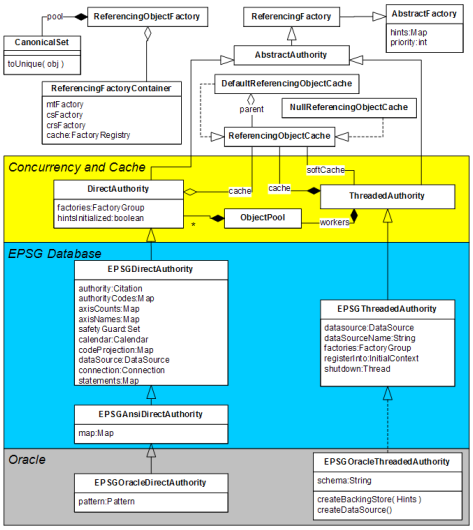
	Dispatch and Adapters
`DefaultAuthorityDecorator`	The default CRSAuthorityFactory used by client code such as the CRS facade
`OrderedAxisAuthorityDecorator`	Decorator often used to reorder axis to agree with the expectations of simple web software
`AllAuthority`	Acts as an "Adapter", making all known crs authorities avaialble for one stop shopping
`ReferencingFactoryFinder`	Uses a FactoryRegistry to manage as singletons all the following ....
	Authority Implementations
`EPSGOracleThreadedAuthority`	A "Builder" is able to convert from epsg code into full `CoordinateReferenceSystem` instances, delegates out work to a pool of `EPSGOracleDirectAuthority` instances.
`AutoAuthority`	A "Builder" that takes hard coded definitions of "AUTO" and "AUTO2" codes and makes them available
`EPSGPropertyFileAuthority`	Used to hoist "extra" epsg codes definitions in common use
	Internals
`ReferencingObjectCache`	A cache used for storing referencing objects
`ObjectPool`	From the apache commons library, used to manage worker lifecycle

`EPSGOracleDirectAuthority`	A "Builder" that uses the definitions provided by the EPSG database loaded into oracle tables

Allowing Multiple Users

`EPSGOracleThreadedAuthority` allows multiple threads, making use of `ReferencingObjectCache` in order to return objects previously constructed and an `ObjectPool` of workers to create new content in the event of a cache miss.

Class Diagram

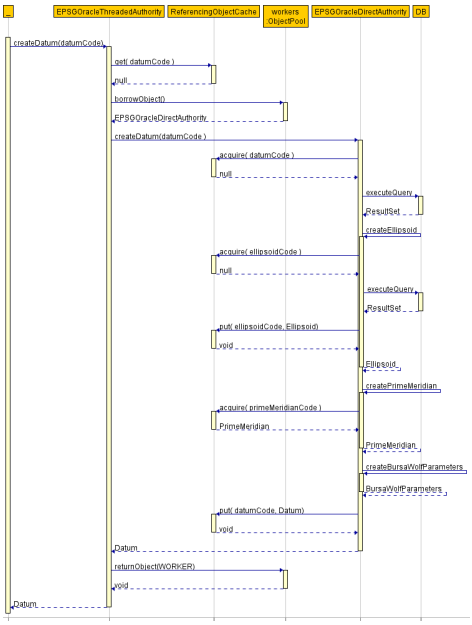


To build compound objects the workers will need to share the cache with the parent.

Class	Theadsafe	
`EPSGOracleThreadedAuthority`	yes	Allows multiple threads
`EPSGOracleDirectAuthority`	yes	All public methods are **synchronized** (allowing class to be used in a standalone fashion
`ReferencingObjectCache`	yes	Allows multiple readers, read/write lock used on individual cache entries

The following sequence diagram shows the behaviour of `EPSGOracleThreadedAuthority` when responding to a **createDatum** request. Initially the requested datum is not in the cache, a worker is retrieved from the `ObjectPool` and used to perform the work. Of interest is the use of the shared `ReferencingObjectCache` to block subsequent workers from duplicating this activity.

Sequence Diagram



Cache Handling

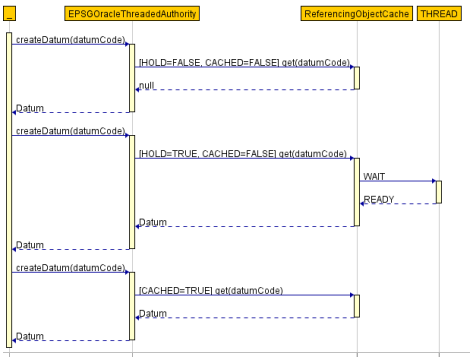
The cache has been isolated into a single class - `ReferencingObjectCache`. This class is responsible for storing strong references to objects already created and released to code outside of the referencing module.

The ReferencingObjectCache stores an internal Map<Obj,Reference> as described in the following table.

Reference	Use
weak	Used by default to store cache contents
strong	Used for frequently used objects up until a configured threshold (default of 50)
placeholder	Placed into the cache to block readers, used to indicate work in progress during object construction

- During a Cache Hit: The entry is looked up in the internal map, if the value is found it is returned. If a soft reference is found the value is extracted if needed (the soft reference may be changed into a strong reference if needed).
- During a Cache Miss: A worker is produced to construct the appropriate object; the worker will reserve the spot in the cache with a placeholder reference, produce the required object and place it into the cache. The placeholder will then be removed and all threads waiting for the value released.
- During a Cache Conflict: More than one worker is released to construct the appropriate object; the first worker will reserve using a placeholder object - and the second will block. When the placeholder is released the second worker will consider itself as having undergone a cache hit and behave as normal.

Sequence Diagram



As noted above the `ReferencingObjectCache` class is thread safe.

Implementation Note - Metadata search via the Find method
The find method makes use of fully created referencing objects (like Datum and CoordinateReferenceSystem) in order to make comparisons using all available meta data. This workflow involves creating (and throwing away) lots of objects; and falls outside of our normal usage patterns. To facilitate this work flow: • A separate **softCache** is maintained - configured to only use weak references. This **softCache** uses the real **cache** as its parent.

Implementation Note - Making good use of Memory
`ReferencingObjectFactory` uses an internal CanonicalSet to prevent more than one referencing object with the same definition being in circulation. The internal pool makes use of weak references.

Connection Issues

`EPSGOracleThreadedAuthority` is the keeper of a `DataSource` which is provided to `EPSGOracleDirectAuthority` workers on construction. The `EPSGOracleDirectAuthority` workers use their **dataSource** to create a **connection** as needed, they will also keep a cache of `PreparedStatements` opened against that connection.

The `ObjectPool` lifecycle methods are implemented allowing `EPSGOracleDirectAuthority` object to be notified when no longer in active use. At this point their `PreparedStatements` and `Connection` can be closed - and reclaimed by the `DataSource`

We will need to make use of a single worker (and use it to satisfy multiple definitions) when implementing the find method.

By providing hints to tune the `ObjectPool` we can allow an application to:

- Ensure that less workers are in play than number of Connections managed by the DataSource (so other oracle modules do not starve)
- Emulate the current 20 min timeout behavior
- Arrive at a compromise for J2EE applications (where a worker can free it's connection the moment it is no longer in constant use)

Documentation Changes

Update Module matrix pages

- Referencing Module
- EPSG-Oracle Plugin

Update User Guide:

- ObjectCache - done
- Update Referencing Developers Guide
  - 0 Referencing Overview
  - 2 Referencing Factories
  - 3 Authorities backed by the EPSG Database

Issue Tracker:

- check related issues to see of problems are affected