LastResourceCommit12

Last Resource Commit optimization

> **Be Careful**
>
> There is one caveat with Last Resource Commit. There is a small chance that a transaction ends up with inconsistent results across participating resources if BTM crashes while a transaction is in-flight. The chance is small but it exists so be careful when using that feature. This is in *no way* a limitation of BTM but of the concept itself.
>
> Please note that if you only intend to run transactions against a single database using Last Resource Commit this scenario is 100% safe, otherwise not.

In theory, only databases supporting XA and providing a `javax.sql.XADataSource` implementation can be used with transaction managers. In practice, there is a way around this limitation.

The Last Resource Commit optimization (sometimes referred to as Last Resource Gambit or Last Agent optimization) allows a single non-XA database to participate in a XA transaction by cleverly ordering the resources.

> **Maximum one non-XA resource**
>
> There can be at most one datasource emulating XA with Last Resource Commit participating in a transaction. If it happens that you're trying to use a second emulating datasource while one has already been used, BTM will throw an exception. Again, this is not a limitation of BTM but of the concept itself.

To enable it, you just have to create a `PoolingDataSource` using the bitronix.tm.resource.jdbc.lrc.LrcXADataSource as the `XADataSource` implementation.

Here's an example of code configuring a HSQLDB datasource:

```
PoolingDataSource myDataSource = new
PoolingDataSource();
myDataSource.setClassName("bitronix.tm.resou
rce.jdbc.lrc.LrcXADataSource");
myDataSource.setUniqueName("hsqldb");
myDataSource.setMaxPoolSize(5);
myDataSource.setAllowLocalTransactions(true)
;
myDataSource.getDriverProperties().setProper
ty("driverClassName",
"org.hsqldb.jdbcDriver");
myDataSource.getDriverProperties().setProper
ty("url", "jdbc:hsqldb:/the/db/path");
myDataSource.getDriverProperties().setProper
ty("user", "sa");
myDataSource.getDriverProperties().setProper
ty("password", "theSaPassword");
```

and the same example viewed as a Resource Loader configuration

```
resource.ds.className=bitronix.tm.resource.jdbc.lrc.LrcXADataSource
resource.ds.uniqueName=hsqldb
resource.ds.maxPoolSize=5
resource.ds.allowLocalTransactions=true
resource.ds.driverProperties.driverClassName=org.hsqldb.jdbcDriver
resource.ds.driverProperties.url=jdbc:hsqldb:/the/db/path
resource.ds.driverProperties.user=sa
resource.ds.driverProperties.password=theSaPassword
```

Mandatory pool settings

The LRC implementation relies on the `useTmJoin` and `deferConnectionRelease` pool properties which must always be **true** (the default value).
Do not change these properties as this could force the transaction manager to reject the resource or worse, cause inconsistencies during commit phase.