WorkingWithOptimisticConcurrencyControl

WorkingWithOptimisticConcurrencyControl

The RDB DAS provides a straightforward mechanism for support of OptimisticConcurrencyControl. To enable OCC a client must designate an integer version column in the configuration (usually via a config xml file). The following example is an example of a cofig file that identifies a version column:

```
<Config xsi:noNamespaceSchemaLocation="http:///org.apache.tuscany.das.rdb/config.xsd" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance">

     <Command name="select book 1" SQL="select * from BOOK where BOOK_ID = 1" kind="Select"/>

     <Table tableName="BOOK">
        <Column columnName="BOOK_ID" primaryKey="true"/>
        <Column columnName="OCC" collision="true" managed="true"/>
     </Table>

   </Config>
```

This line in the config file _<Column columnName="OCC" collision="true" managed="true"/>_designates the column named "OCC" as the version column and also sets "managed" to true. The "managed" attribute has to do with how the version column is changed. In order for OCC to work, some agent must change the value of the version column each time the row is modified. A new value in the version column is what tells the DAS that the row has been modified. If a version column is "managed" then the DAS is responsible for incrementing the value of the column each time some column in the row is changed. The other option is "managed= false" and in this case, the DAS will assume some other agent is responsible for updating the version column. This other agent is likely to be the client application itself or the database via a trigger or some other mechanism.

When applying changes made to a graph (see ChangeSummaryProcessing), the DAS will throw an exception if any table row has been modified since the row was read. For example, if a DAS client reads a row representing a specific Customer and then tries to update that Customer, the DAS will throw an exception if the Customer has already been modified by some other agent. The following example illustrates applying changes and checking for a collision failure:

```
DAS das = DAS.FACTORY.createDAS(getConfig("ManagedBooksConfig.xml"), getConnection());
   // Read a book instance
   Command select = das.getCommand("select book 1");
   DataObject root = select.executeQuery();
   DataObject book = root.getDataObject("BOOK[1]");
   book.setInt("QUANTITY", 2);

   // Try to apply changes and catch potential update collision
   try {
      das.applyChanges(root);
   } catch (RuntimeException ex) {
      if (!ex.getMessage().equals("An update collision occurred")) {
          recover();
      }
   }
```