WorkingWithPaging

WorkingWithPaging

The RDB DAS provides a mechanism for working with the result of large queries one chunk (page) at a time. This capability is modeled after the behavior of web search engines. For example, if I Google "RDB DAS" I am presented with a page representing the first ten references. I am also told that there were close to 500,000 matches and I am given the opportunity to go to the next page or directly to pages two through ten. After moving from the first page I am also provided the opportunity to move back to some previous page.

The DAS "Pager" is a wrapper around a provided read command and provides the APIs for paging. Here is the Pager interface:

```
public interface Pager {

    /**
     * Get the next page of data
     */
    DataObject next();

    /**
     * Get the page prior to the last page returned
     */
    DataObject previous();

    /**
     * Return a specific identified page.
     */
    DataObject getPage(int page);

}
```

A Pager is instantiated on a given read Command and the constructor is used to specify the desired page size (number of database rows per page). The following example illustrates construction and use of a pager:

```
DAS das = DAS.FACTORY.createDAS(getConnection());
   // Build command to read all customers
   Command custCommand = das.createCommand("select * from CUSTOMER order by ID");

   // Create a pager with the command
   Pager pager = new PagerImpl(custCommand, 2);

   // Get and work with first page
   DataObject root = pager.next();
   DataObject customer1 = root.getDataObject("CUSTOMER[1]");
   DataObject customer2 = root.getDataObject("CUSTOMER[2]");

   // Get and work with the second page
   root = pager.next();
   customer1 = root.getDataObject("CUSTOMER[1]");
   customer2 = root.getDataObject("CUSTOMER[2]");

   // First page again
   root = pager.previous();
   customer1 = root.getDataObject("CUSTOMER[1]");
   customer2 = root.getDataObject("CUSTOMER[2]");
```

The current Pager specification and implementation are very preliminary and will likely change in significant ways! For instance, the pager is intended to work in completely disconnected scenarios so there should be no connection context maintained between invocations of the pager. Currently the Pager references a Command that, in turn, references a connection (oops!). Some work must be done to make the pager lightweight as it will likely be stored in session.

We probably should also create a Factory rather than using the impl constructor.