justinsuen / cs186
forked from scyantist/course

Branch: master cs186 / hw3 / src / main / java / edu / berkeley / cs186 / database / query / PNLJOperator.java Find file Copy path

John Wilkey Implement HW3 64c79ee on Oct 21, 2017

1 contributor

169 lines (147 sloc) 5.88 KB Raw Blame History

```
package edu.berkeley.cs186.database.query;

import edu.berkeley.cs186.database.Database;
import edu.berkeley.cs186.database.DatabaseException;
import edu.berkeley.cs186.database.common.BacktrackingIterator;
import edu.berkeley.cs186.database.databox.DataBox;
import edu.berkeley.cs186.database.io.Page;
import edu.berkeley.cs186.database.table.Record;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.NoSuchElementException;

public class PNLJOperator extends JoinOperator {

  public PNLJOperator(QueryOperator leftSource,
                      QueryOperator rightSource,
                      String leftColumnName,
                      String rightColumnName,
                      Database.Transaction transaction) throws QueryPlanException, DatabaseException {
    super(leftSource,
          rightSource,
          leftColumnName,
          rightColumnName,
          transaction,
          JoinType.PNLJ);
  }

  public Iterator<Record> iterator() throws QueryPlanException, DatabaseException {
    return new PNLJIterator();
  }


  /**
   * An implementation of Iterator that provides an iterator interface for this operator.
   */
  private class PNLJIterator extends JoinIterator {
    private final BacktrackingIterator<Page> LPIter;
    private BacktrackingIterator<Page> RPIter;
    private Page currentLeftPage;
    private Page currentRightPage;
    private BacktrackingIterator<Record> LBIter;
```

```java
    private BacktrackingIterator<Record> RBIter;
    private Record leftRecord;
    private Record nextRecord;

    public PNLJIterator() throws QueryPlanException, DatabaseException {
      super();
      LPIter = getPageIterator(getLeftTableName());
      RPIter = getPageIterator(getRightTableName());

      // Consume header pages
      RPIter.next();
      LPIter.next();

      currentLeftPage = LPIter.next();
      currentRightPage = RPIter.next();

      LBIter = getBlockIterator(getLeftTableName(),
                                new Page[]{currentLeftPage});
    }

    /**
     * We're effectively doing the following in the hasNext() function:
     *
     * for LeftPage in LeftPageIter:
     *     for RightPage in RightPagerIter:
     *         For LeftRecord in LeftPage:
     *             For RightRecord in RightPage:
     *                 check(LeftRecord, RightRecord)
     *
     * If an iterator lower in the nested loops becomes exhausted, we check
     * if it's parent has more elements. If it does, we just restart the
     * iterator and advance the parent to the next element. If not, we check
     * the parent's parent for the same and repeat. If we make it to the top
     * of the nested loop and run out of elements, that means LeftPageIter
     * is exhausted, and we're done.
     */
    public boolean hasNext() {
      if (nextRecord != null) {
        return true;
      }

      try {
        while (true) {
          if (leftRecord == null) {
            if (LBIter.hasNext()) {
              // If left page still has records to give, then just get the
              // next one and restart right block iter.

              leftRecord = LBIter.next();
              RBIter = getBlockIterator(getRightTableName(),
                                        new Page[]{currentRightPage});
            } else {
              // Current left page is exhausted. Restart it with the next
              // right page (if there is one).

              if (!RPIter.hasNext()) {
                // Right page relation is exhausted. Need to restart it with
                // LPIter on the next page (if there is one).

                if (LPIter.hasNext()) {
                  currentLeftPage = LPIter.next();
                  LBIter = getBlockIterator(getLeftTableName(),
                                            new Page[]{currentLeftPage});
                  assert LBIter.hasNext() : "Need to hasNext() first.";
                  leftRecord = LBIter.next();
                } else {
                  // Outermost relation exhausted so we're done.
              return false;
            }

            RPIter = getPageIterator(getRightTableName()); // Restart RP
            RPIter.next(); // Consume header page
          } else {
            LBIter = getBlockIterator(getLeftTableName(),
                                      new Page[]{currentLeftPage});
            assert LBIter.hasNext() : "LBIter degenerate";
            leftRecord = LBIter.next();
          }

          currentRightPage = RPIter.next();
          RBIter = getBlockIterator(getRightTableName(),
                                    new Page[]{currentRightPage});
        }
      }
      while (RBIter.hasNext()) {
        Record rightRecord = RBIter.next();
        DataBox leftJoinValue = leftRecord.getValues().get(getLeftColumnIndex());
        DataBox rightJoinValue = rightRecord.getValues().get(getRightColumnIndex());
        if (leftJoinValue.equals(rightJoinValue)) {
          List<DataBox> leftValues = new ArrayList<>(leftRecord.getValues());
          List<DataBox> rightValues = new ArrayList<>(rightRecord.getValues());
          leftValues.addAll(rightValues);
          nextRecord = new Record(leftValues);
          return true;
        }
      }
      leftRecord = null;
    }
  } catch (DatabaseException e) {
    System.err.println("Caught database error " + e.getMessage());
    return false;
  }
}

/**
 * Yields the next record of this iterator.
 *
 * @return the next Record
 * @throws NoSuchElementException if there are no more Records to yield
 */
public Record next() {
  if (nextRecord != null) {
    Record out = nextRecord;
    nextRecord = null;
    return out;
  }

  throw new NoSuchElementException("next() on empty iterator");
}

public void remove() {
  throw new UnsupportedOperationException();
}
  }
}
```