justinsuen / cs186
forked from scyantist/course

sjyk Added in homework hw4 25c1d7f on Oct 19, 2017

1 contributor

136 lines (119 sloc) 4.36 KB Raw Blame History

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

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

import edu.berkeley.cs186.database.Database;
import edu.berkeley.cs186.database.DatabaseException;
import edu.berkeley.cs186.database.databox.DataBox;
import edu.berkeley.cs186.database.table.Record;

public class SNLJOperator extends JoinOperator {
  private QueryOperator leftSource;
  private QueryOperator rightSource;
  private int leftColumnIndex;
  private int rightColumnIndex;
  private String leftColumnName;
  private String rightColumnName;
  private Database.Transaction transaction;

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

    this.stats = this.estimateStats();
    this.cost = this.estimateIOCost();

    this.leftSource = getLeftSource();
    this.rightSource = getRightSource();
    this.leftColumnIndex = getLeftColumnIndex();
    this.rightColumnIndex = getRightColumnIndex();
    this.leftColumnName = getLeftColumnName();
    this.rightColumnName = getRightColumnName();
    this.transaction = getTransaction();
  }

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

  public int estimateIOCost() throws QueryPlanException {
        int numLeftRecords = getLeftSource().getStats().getNumRecords();

        int numRightPages = getRightSource().getStats().getNumPages();
        int numLeftPages = getLeftSource().getStats().getNumPages();

        return numLeftRecords * numRightPages + numLeftPages;
  }


  /**
   * An implementation of Iterator that provides an iterator interface for this operator.
   */
  private class SNLJIterator implements Iterator<Record> {
    private Iterator<Record> leftIterator;
    private Iterator<Record> rightIterator;
    private Record leftRecord;
    private Record nextRecord;

    public SNLJIterator() throws QueryPlanException, DatabaseException {
      this.leftIterator = SNLJOperator.this.getLeftSource().iterator();
      this.rightIterator = null;
      this.leftRecord = null;
      this.nextRecord = null;
    }

    /**
     * Checks if there are more record(s) to yield
     *
     * @return true if this iterator has another record to yield, otherwise false
     */
    public boolean hasNext() {
      if (this.nextRecord != null) {
        return true;
      }
      while (true) {
        if (this.leftRecord == null) {
          if (this.leftIterator.hasNext()) {
            this.leftRecord = this.leftIterator.next();
            try {
              this.rightIterator = SNLJOperator.this.getRightSource().iterator();
            } catch (QueryPlanException q) {
              return false;
            } catch (DatabaseException e) {
              return false;
            }
          } else {
            return false;
          }
        }
        while (this.rightIterator.hasNext()) {
          Record rightRecord = this.rightIterator.next();
          DataBox leftJoinValue = this.leftRecord.getValues().get(SNLJOperator.this.getLeftColumnIndex());
          DataBox rightJoinValue = rightRecord.getValues().get(SNLJOperator.this.getRightColumnIndex());
          if (leftJoinValue.equals(rightJoinValue)) {
            List<DataBox> leftValues = new ArrayList<DataBox>(this.leftRecord.getValues());
            List<DataBox> rightValues = new ArrayList<DataBox>(rightRecord.getValues());
            leftValues.addAll(rightValues);
            this.nextRecord = new Record(leftValues);
            return true;
          }
          }
          this.leftRecord = null;
        }
      }

      /**
       * 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 (this.hasNext()) {
          Record r = this.nextRecord;
          this.nextRecord = null;
          return r;
        }
        throw new NoSuchElementException();
      }

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