CSCI 3357: Database System Implementation

Homework Assignment 6

Due Wednesday, October 23

This assignment asks you to make two changes to the SimpleDB record manager. To simplify debugging, I strongly recommend that you don't begin the second one until the first is working.

1. SimpleDB only knows how to read files in the forward direction. Add the following methods to the class `TableScan`:

- the method `previous()`, which moves to the previous record in the file and returns false if there is no such record;
- the method `afterLast()`, which positions the current record to be after the last record in the file.

To implement `previous`, you will need to search a record page backwards. To implement `afterLast`, you will need to position the current slot at the end of the page. For ease of grading, I would like you to do this by implementing the following methods in `RecordPage`:

- the method `slots()`, which returns the number of slots in the page;
- the method `nextBefore(int slot)`, which returns the next used slot before the specified one, or -1 if no such slot exists.

You are, of course, free to implement other private methods in either class.

2. Revise the SimpleDB record manager to handle null field values. To do so, you must add the following two public methods to `TableScan`:

- the method `setNull(fldname)`, which sets the value of the specified field in the current record to null.
- the method `isNull(fldname)`, which determines if the value of the specified field in the current record is null.

Note that the `isNull` method is the only way a client can know if a value is null. Suppose that you access the value of a integer field, say, by doing `ts.getInt("GradYear")`. There is no integer whose value can be treated as a null. Similarly, there is no special string value that can be treated as a null. (The string "null" is a legitimate non-null string.) That is why you need a special method to tell you whether the value is actually null, regardless of what `getInt` or `getString` returns. `TableScan` cannot directly handle null values. Instead, it must ask `RecordPage` to do the work. That is, its `isNull` and `setNull` methods should be implemented as follows:

```
public boolean isNull(String fldname) {
   return rp.isNull(currentslot, fldname);
}
```

```
   public void setNull(String fldname) {
      rp.setNull(currentslot, fldname);
   }
```

So your job is to implement `isNull` and `setNull` in `RecordPage`. How to set a field value to null? Since it is unreasonable to use a particular integer or string value to denote a null, you should use a one-bit flag. In particular, say that a record contains N fields. Suppose that you store N additional bits with each record and assign the value of the i^{th} bit to be 1 if the value of the i^{th} field is null and 0 if it is non-null. If you assume that N<32, then you can use the `EMPTY/INUSE` integer for this purpose. Bit 0 of this integer (counting from the right) denotes empty/inuse, as before. But now you can use the other bits to hold null-value information.

You will need to modify the `Layout` constructors so that they assign a bit position to each field in the record. (So for example if the schema has fields "A" and "B", then "A" might be assigned position 1 and "B" position 2.) Be careful to perform this assignment the same way in both constructors. Your `Layout` class should also implement the following new method, which will allow the record page to determine the bit position of any field:

```
      public int bitPosition(String fldname);
```

The `setNull` and `isNull` methods of `RecordPage` will need to get and set individual bits of the empty/inuse integer. Since not all of you have learned how to do this, I have written the following two methods for you:

- The method *getBitVal* returns the value of a specified bit (from the right, and beginning at 0) of a given integer.
- The method *setBitVal* returns the integer you get by setting a specified bit (from the right and beginning at 0) of a specified integer to a specified flag (which will be either 0 or 1).

```
      private int getBitVal(int val, int bitpos) {
           return (val >> bitpos) % 2;
      }

      private int setBitVal(int val, int bitpos, int flag) {
           int mask = (1 << bitpos);
           if (flag == 0)
                return val & ~mask;
           else
                return val | mask;
      }
```

You will also need to make changes to other methods in `RecordPage`. Look over the code carefully. All methods that access the empty/inuse flag will need to be adjusted. In addition, please ensure that the `format` method sets the fields of each record to be non-null, and that the `setInt` and `setString` methods also set their field to non-null.