

Report of Labs

Apr 26, 2020

Report of Lab2

- Describe any design decisions you made, including your choice of page eviction policy. Describe briefly your insertion and deletion methods in B+ tree. Describe your idea in solving the bonus exercise (if it applies).

My page eviction policy is LRU(Least Recently Used).

To insert or delete a tuple in B+ tree, firstly I find the leaf page to insert or delete by comparing the key in entries recursively.

Having found the leaf page, I find the slot of the tuple, and insert or delete the tuple.

Finally, I have to maintain the B+ tree by stealing or merging the pages recursively.

- Discuss and justify any changes you made to the API.

None.

- Describe any missing or incomplete elements of your code.

None.

- Describe how long you spent on the lab, and whether there was anything you found particularly difficult or confusing.

7 days. The most difficult part is maintaining the B+ tree. It's a tedious work but worth doing.

Report of Lab1

- Describe any design decisions you made. These may be minimal for Lab 1.

Create `HeapFileIterator` class to implement iterator for `HeapFile`.

Create `HeapPageIterator` class to implement iterator for `HeapPage`.

- Discuss and justify any changes you made to the API.

None.

- Describe any missing or incomplete elements of your code.

– In `Catalog.java`

* Create two private `ConcurrentHashMap` members in `Catalog`.

* Implement `Catalog`, `addTable`, `getTableId`, `getTupleDesc`, `getDatabaseFile`, `getPrimaryKey`, `tableIdIterator`, `getTableName` and `clear` functions.

– In `HeapFile.java`

* Create private `File` and `TupleDesc` members.

* Implement `HeapFile`, `getFile`, `getId` and `getTupleDesc` functions.

- * Implement `readPage` function. Read a page through a page ID by reading specific bytes on the page.
- * Implement `numPages` function. The number of pages equals $\text{ceiling}(\text{file length} / \text{page size})$.
- In `HeapPage.java`
 - * Create private `File` and `TupleDesc` members.
 - * Implement `getNumTuples`, `getHeaderSize`, `getId` and `iterator` functions.
 - * Implement `getNumEmptySlots` function. To get the number of empty slots, I go through every slot and sum up every empty slot.
 - * Implement `isSlotUsed` function. The `ith` bit of `header` indicates whether `ith` slot is filled.
- In `HeapPageId.java`
 - * Implement `HeapPageId`, `getTableId`, `pageNumber` `hashCode` and `equals` functions.
- In `RecordId.java`
 - * Implement `RecordId`, `equals`, and `hashCode` functions.
- In `SeqScan.java`
 - * Create class members `transId`, `tableId`, `tableAlias`, `file`, `iterator`.
 - * Implement `SeqScan`, `getAlias`, `open`, `getTupleDesc`, `hasNext`, `next`, `close` and `rewind` functions.
- In `Tuple.java`
 - * Create class members `TupleDesc`, `fields`, `rid`.
 - * Implement `Tuple`, `getTupleDesc`, `getRecordId`, `setRecordId`, `setField`, `getField`, `toString`, `fields` and `resetTupleDesc`.
- In `TupleDesc.java`
 - * Implement `TupleDesc`, `numFields`, `getFieldName`, `getFieldType`, `fieldNameToIndex`, `getSize`, `merge`, `equals` and `toString`.
- Describe how long you spent on the lab, and whether there was anything you found particularly difficult or confusing.

Three days.

The most difficult part is `read` function in `HeapFile` and `HeapPage` class.