SUTD 2021 50.043 SimpleDB Project Part 1 Writeup Report Document

James Raphael Tiovalen / 1004555

Implementation Description

For part 1 of the project, I simply implemented most of the skeleton methods/functions and classes specified by the handout. I followed the implementation guide as laid out by the exercises closely, and hence, there are very few deviations from the intended path.

Several design decisions that I have made include:

- Implementing the `hashCode()` methods for `src/java/simpledb/storage/HeapPageId.java` and `src/java/simpledb/storage/RecordId.java`. I have decided to follow other similar classes that also implement a custom `hashCode()` implementation, which is by using `Objects.hash()`. I commented out a possible alternative implementation that should also work, which was inspired by the Effective Java book written by Joshua Bloch.
- Implementing a custom data class/structure in `src/java/simpledb/common/Table.java` for use in `src/java/simpledb/common/Catalog.java`. This way, we can have a much neater codebase architecture and organization. An alternative would be to store multiple `ConcurrentHashMap`s in `src/java/simpledb/common/Catalog.java`, which will be quite messy.
- I used `ConcurrentHashMap` instead of `HashMap` for `src/java/simpledb/common/Catalog.java` and `src/java/simpledb/storage/BufferPool.java` to allow said classes to handle the possibility that multiple threads are trying to access and use the database, which might come in handy in the future.

Several challenges that I have faced include:

- Floating point and integer rounding errors for the `getNumTuples()` and `getHeaderSize()` methods in the `src/java/simpledb/storage/HeapPage.java` file. Using integer values instead of double values caused the `testSmall()` unit test case of the `ScanTest` system test to fail (due to integer underflow/overflow).
- Implementing `src/java/simpledb/storage/HeapFileIterator.java` for use in `src/java/simpledb/storage/HeapFile.java`. The current `HeapFileIterator` extends the `AbstractDbFileIterator` abstract class (acquired from `src/java/simpledb/storage/AbstractDbFileIterator.java`) to allow for more uniformity in terms of implementation across the codebase, as well as ensuring that the implementation would most likely work (instead of using a self-constructed implementation from scratch that require a lot of debugging time).

I did not change the provided API at all. All the custom, additional classes are designed around the current API to support it, instead of subverting or redirecting it.

There are no missing or incomplete elements of my code, assuming that all that is being considered is for part 1. Any other extra additional methods will be implemented in future parts, since this provides ease of debugging and separation between different functionalities as the project timeline progresses.