Lab 1 Report

Exercise 1: Design Decisions

Tuple Class

- Attributes: Added attributes `TupleDesc` (instance of Tuple class), `fields` (to store field data), and `recordId` (to store tuple's disk location).
- Implementation: Followed the provided comments for function implementation, including initialization and methods to get or modify field data.

TupleDesc Class

- Attributes: Defined a `tdItems` list to store `TDItem` objects.
- Implementation: Introduced a private helper function `initializeTdItems` to avoid code duplication during initialization. Completed functions as per comments. Utilized Java 8 stream features to improve `equals` and `getSize` methods.

Exercise 2: Design Decisions

Table Class

- Implemented a `Table` class to store tables in the database.
- Attributes: Designed to include file, primary key name, and table name.
- Functions: Created methods for querying and setting table properties. Added checks for duplicate table names in the add table function.

Exercise 3: Design Decisions

BufferPool

- Implemented a hashmap to store mappings from `PageId` to `Page`.
- Features: Added functionality to read from the disk if the page is not in the `BufferPool`. Included a limit for the maximum number of pages the pool can store.

Exercise 4: Design Decisions

HeapPageId.java

- Attributes: Defined `tableId` (table number) and `pageNo` (page number).
- Implementation: Completed the code as per comments. Used the `Object` module for hash calculation.

RecordId.java

- Similar implementation to `HeapPageId`.

HeapPage.java

- Calculations: Implemented formulas for calculating the number of tuples and header size.
- Slot Checking: Implemented methods to count empty slots and check if a slot is occupied by checking the corresponding bit in the bitmap.

Exercise 5: Design Decisions

Implementation

- Attributes: Defined `f` (file) and `td` (tuple) in the class.
- `readPage`: Retrieves a specific page from the associated database file based on `PageId`.
- Iterator: Defined `currentPageNo` and `currentTupleIterator`. Implemented `open`, `hasNext`, `next`, `rewind`, and `close` functions for iteration. Included `getPageTupleIterator`, a helper method to get a tuple iterator for a specified page from the `BufferPool`.

Exercise 6: Design Decisions

Implementation

- Attributes: Defined `tid`, `tableId`, `tableAlias`, and a tuple iterator `iterator`.
- Iterator Methods: Directly utilized methods from the `iterator` designed in Exercise 5.

Time Spent & Challenges:

- Spent approximately 5 days on the lab.
- The most challenging part was implementing the iterator methods in Exercise 5, particularly managing the transition between different pages and ensuring the iterator correctly handles end-of-page scenarios. Also, optimizing the `HeapPage` calculations in Exercise 4 to efficiently compute the number of tuples and header size required a deep understanding of the underlying data structure, which was initially confusing.