I. Experimental Environment

- Ubuntu 16.04
- JDK 1.8
- Ant 1.9.6

II. Experimental Procedure

Lab1 primarily involves implementing the overall architecture of a database, which enhances the understanding of the organization of databases after completion.

Lab1 covers the following components: Tuple, TupleDesc (tuple descriptor), Catalog, BufferPool, HeapPage, HeapFile (file on disk), and SeqScan (sequential full table scan).

Exercise 1: Fields and Tuples

A Tuple represents a record in a table and contains three attributes: td, rid, and fields, among other elements like constructors, getter methods, and iterators.

A Field contains the specific content of each field. TupleDesc describes the schema of each field, including field types and names, represented by TDItem in the code to describe multiple field descriptions.

It's important to ensure inputs are valid when implementing functions like equals.

Exercise 2: Catalog

The Catalog stores the mapping of table IDs to their names. I defined a Table class to hold information about a table, and a Catalog contains multiple tables.

Exercise 3: BufferPool

The BufferPool is responsible for caching pages recently read from the disk in memory. This allows for quicker data retrieval directly from memory in subsequent queries,
faster than reading from the disk. Lab1 does not require implementing a replacement strategy, mainly focusing on the getPage method. During the experiment, the BufferPool is checked using pid to determine if the data page is present;
if so, it is returned directly, if not, it is retrieved from the disk and stored in the BufferPool. If the cache exceeds its limit, an exception is thrown.

Exercise 4: HeapFile Access Method

The access method reads database data from disk files organized in a specific format. A HeapFile is divided into several HeapPages, each consisting of a fixed array of bytes,
including a header and multiple slots, each slot holding a Tuple. Each page includes a bitmap to indicate whether the i-th slot is occupied.

Besides basic properties, the experiment involves calculating how many tuples a page can store and how many bytes the header occupies, as already provided in the readme.

The overall implementation is straightforward. In subsequent operations on data pages, this information is used to determine if a slot already stores a tuple. It also involves implementing an iterator.

HeapPageId stores the relationship between HeapFile and HeapPage, recording the tableId and page number of the Page. RecordId uniquely identifies a tuple, noting the PageId and tuple index where the tuple is located.

Exercise 5: HeapFile
A HeapFile is an implementation of a DFile, randomly stored on disk. The experiment requires implementing readPage, which involves calculating the file's offset to determine the PageId and then reading data from the disk.
An iterator is also implemented to iterate through all the tuples contained in all pages. To implement this iterator, I built on the basis of DbFileIterator.This involves the creation of an internal class, HeapFileIterator.

Exercise 6: Operators
The Operators here perform a full table scan, primarily based on the iterator implemented in the previous exercise.
The SeqScan implemented supports aliases for the tables being scanned, allowing for modification of each field's name by prefixing it with the alias.

A Simple Query
First, create a file named `some_data_file.txt`, then use `SimpleDb.class` to compile it into a .dat file.
Finally, use the code provided in the lab to test the .dat file. The code adds the table stored in `some_data_file.dat` to the catalog, then performs a scan and outputs the contents of `some_data_file`.

Finish Time
It took about three days to complete Lab1. The initial challenges involved setting up the environment, learning Java syntax, learning how to use Ant, and configuring and using Eclipse. Additionally, it was necessary to understand the flow of the experiment and the structure of the database, as well as to grasp the general framework that had been previously developed by others. 以下是翻译的英文版本：
I. Experimental Environment
- Ubuntu 16.04
- JDK 1.8
- Ant 1.9.6
II. Experimental Procedure
Lab1 primarily involves implementing the overall architecture of a database, which enhances the understanding of the organization of databases after completion.
Lab1 covers the following components: Tuple, TupleDesc (tuple descriptor), Catalog, BufferPool, HeapPage, HeapFile (file on disk), and SeqScan (sequential full table scan).

Exercise 1: Fields and Tuples
A Tuple represents a record in a table and contains three attributes: td, rid, and fields, among other elements like constructors, getter methods, and iterators.
A Field contains the specific content of each field. TupleDesc describes the schema of each field, including field types and names, represented by TDItem in the code to describe multiple field descriptions.

It's important to ensure inputs are valid when implementing functions like equals.

Exercise 2: Catalog
The Catalog stores the mapping of table IDs to their names. I defined a Table class to hold information about a table, and a Catalog contains multiple tables.

Exercise 3: BufferPool
The BufferPool is responsible for caching pages recently read from the disk in memory. This allows for quicker data retrieval directly from memory in subsequent queries,
faster than reading from the disk. Lab1 does not require implementing a replacement strategy, mainly focusing on the getPage method. During the experiment, the BufferPool is checked using pid to determine if the data page is present;
if so, it is returned directly, if not, it is retrieved from the disk and stored in the BufferPool. If the cache exceeds its limit, an exception is thrown.

Exercise 4: HeapFile Access Method
The access method reads database data from disk files organized in a specific format. A HeapFile is divided into several HeapPages, each consisting of a fixed array of bytes,
including a header and multiple slots, each slot holding a Tuple. Each page includes a bitmap to indicate whether the i-th slot is occupied.
Besides basic properties, the experiment involves calculating how many tuples a page can store and how many bytes the header occupies, as already provided in the readme.
The overall implementation is straightforward. In subsequent operations on data pages, this information is used to determine if a slot already stores a tuple. It also involves implementing an iterator.
HeapPageId stores the relationship between HeapFile and HeapPage, recording the tableId and page number of the Page. RecordId uniquely identifies a tuple, noting the PageId and tuple index where the tuple is located.

Exercise 5: HeapFile
A HeapFile is an implementation of a DFile, randomly stored on disk. The experiment requires implementing readPage, which involves calculating the file's offset to determine the PageId and then reading data from the disk.
An iterator is also implemented to iterate through all the tuples contained in all pages. To implement this iterator, I built on the basis of DbFileIterator.This involves the creation of an internal class, HeapFileIterator.

Exercise 6: Operators
The Operators here perform a full table scan, primarily based on the iterator implemented in the previous exercise.
The SeqScan implemented supports aliases for the tables being scanned, allowing for modification of each field's name by prefixing it with the alias.

A Simple Query

First, create a file named `some_data_file.txt`, then use `SimpleDb.class` to compile it into a .dat file.
Finally, use the code provided in the lab to test the .dat file. The code adds the table stored in `some_data_file.dat` to the catalog, then performs a scan and outputs the contents of `some_data_file`.

Finish Time

It took about three days to complete Lab1. The initial challenges involved setting up the environment, learning Java syntax, learning how to use Ant, and configuring and using Eclipse. Additionally, it was necessary to understand the flow of the experiment and the structure of the database, as well as to grasp the general framework that had been previously developed by others.