Database Systems

SimpleDB Overivew

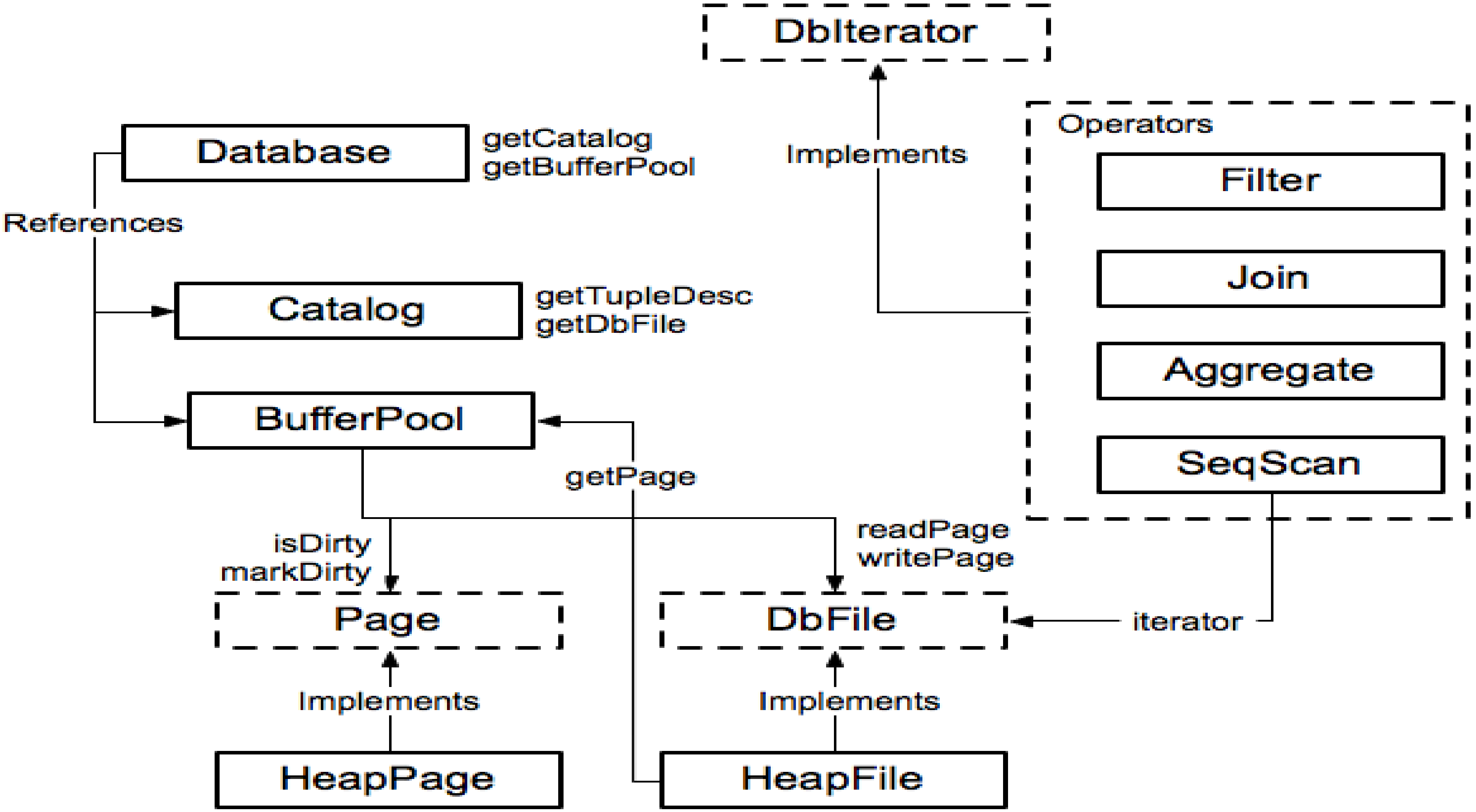
Feifei Li

University of Utah

What is SimpleDB?

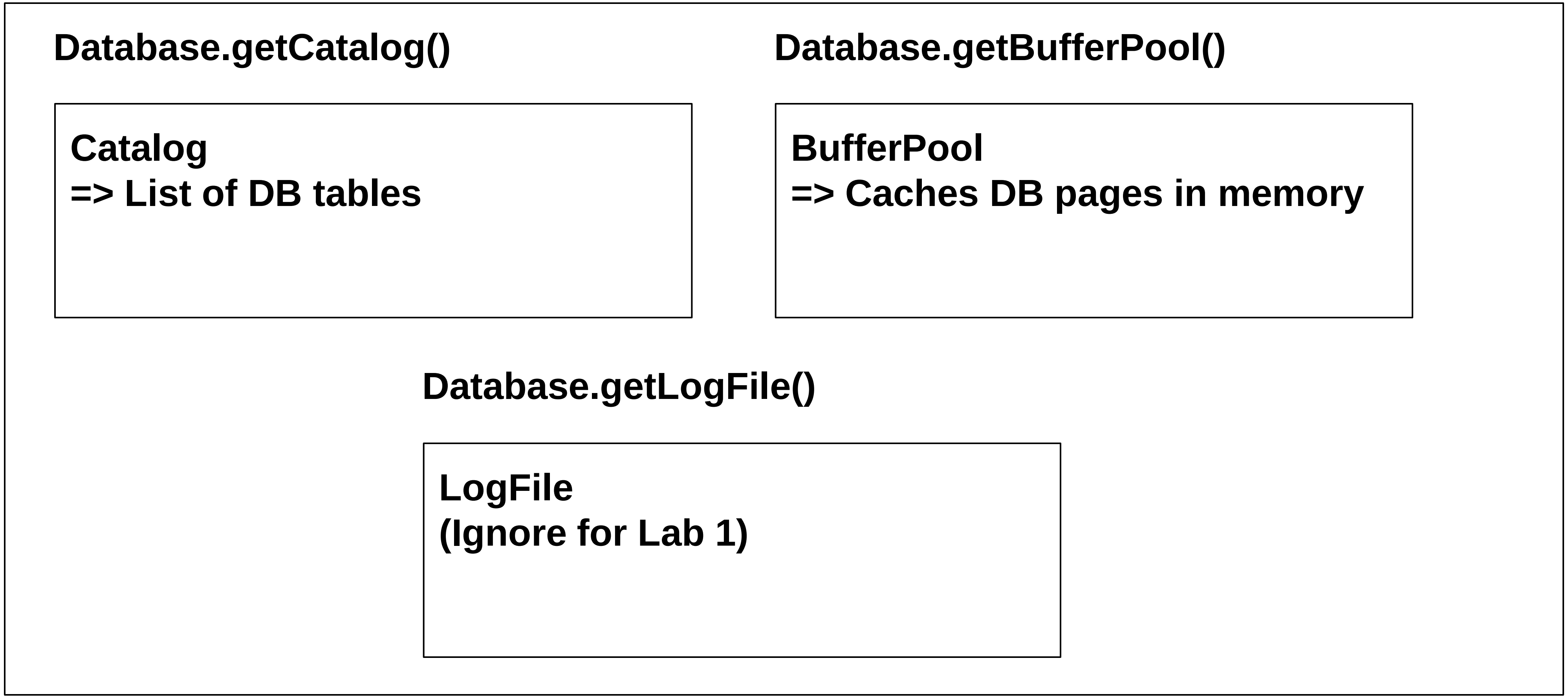
- A basic database system developed by Sam Madden and others from MIT
- SQL Front-end (Provided for later labs)
  - Heap files (Lab 1)
  - Buffer Pool (Labs 1-6)
  - Basic Operators (Labs 1 & 2)
    - Scan, Filter, JOIN, Aggregate
  - B-Tree Indexes (Lab 3)
  - Transactions (Lab 4)
  - Query optimizer (Lab 5)
  - Recovery (Lab 6)
- Javadoc is your friend!

Module Diagram



Database

Singleton Database:



Database.getCatalog()

Catalog
=> List of DB tables

Database.getBufferPool()

BufferPool
=> Caches DB pages in memory

Database.getLogFile()

LogFile
(Ignore for Lab 1)

Catalog

Catalog:

DbFile ID	Table
001	Table1
002	Table2
003	Table3

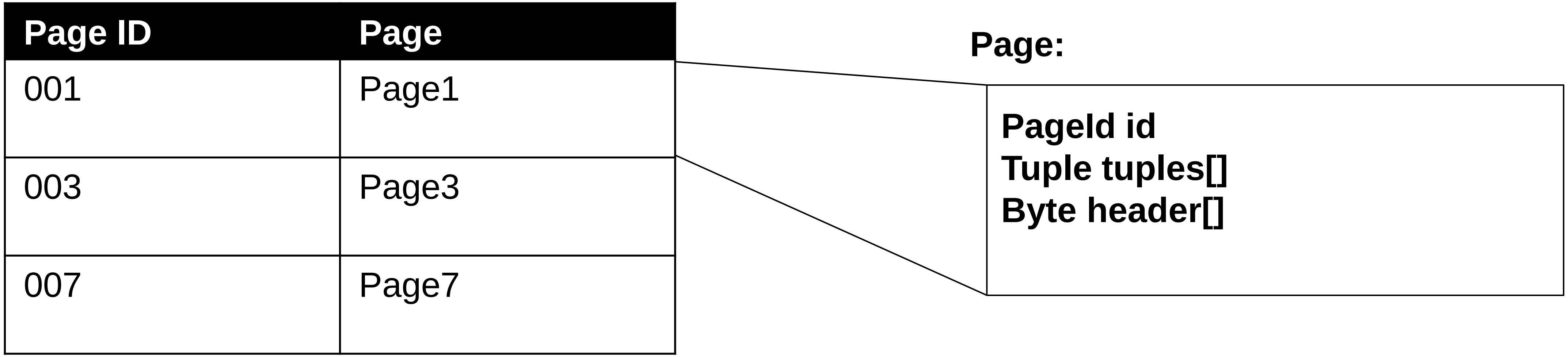
Table:

DbFile file
String name
String primary_key

=> Stores a list of all tables in the database

BufferPool

Buffer Pool:



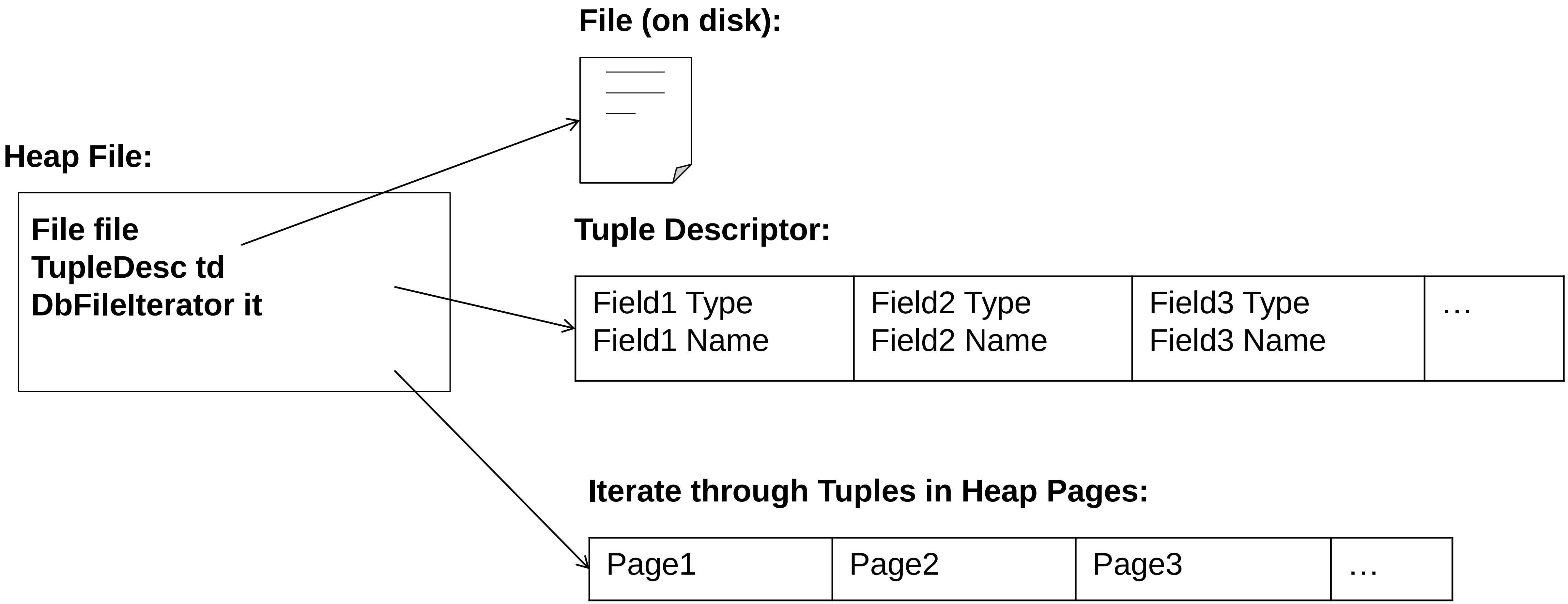
Page ID	Page
001	Page1
003	Page3
007	Page7

Page:

PageId id
Tuple tuples[]
Byte header[]

=> Caches recently accessed database pages in memory

HeapFile (Implements DbFile)



File (on disk):

Heap File:

File file
TupleDesc td
DbFileIterator it

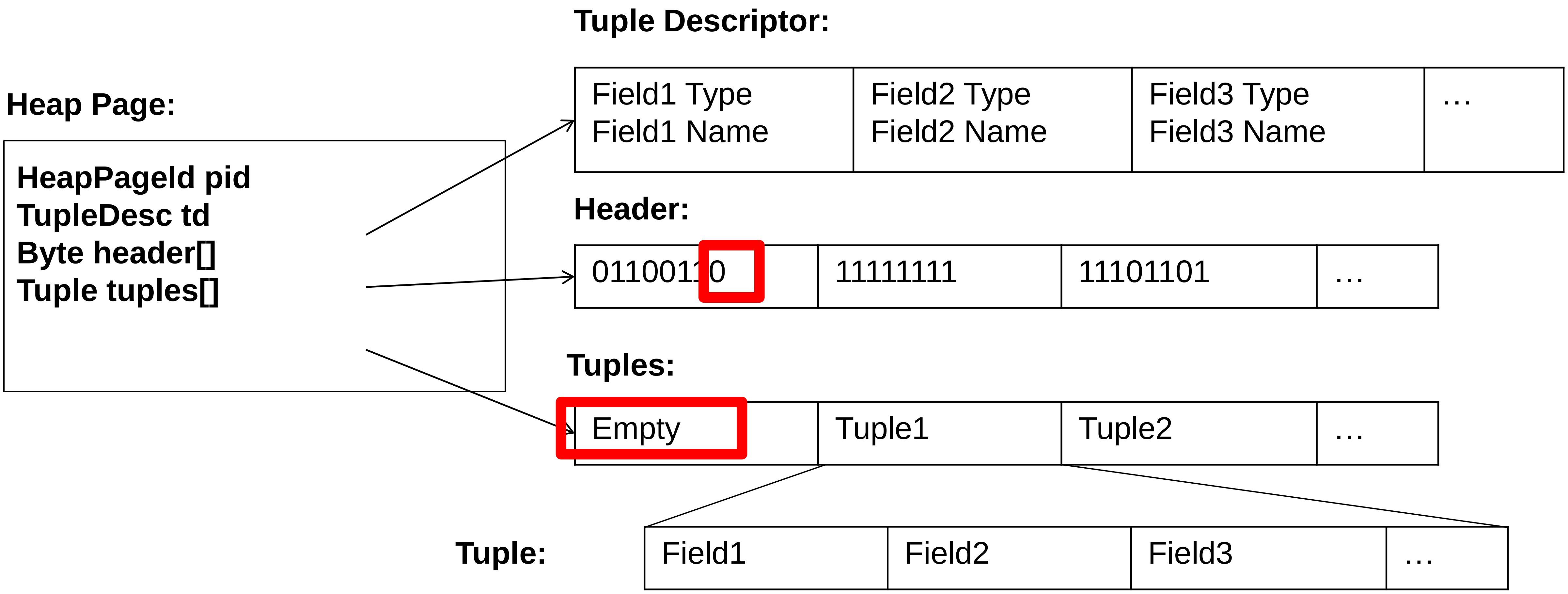
Tuple Descriptor:

Field1 Type Field1 Name	Field2 Type Field2 Name	Field3 Type Field3 Name	…

Iterate through Tuples in Heap Pages:

Page1	Page2	Page3	…

HeapPage (Implements Page)



Tuple Descriptor:

Field1 Type Field1 Name	Field2 Type Field2 Name	Field3 Type Field3 Name	…

Heap Page:

HeapPageId pid
TupleDesc td
Byte header[]
Tuple tuples[]

Header:

01100110	11111111	11101101	…

Tuples:

Empty	Tuple1	Tuple2	…

Tuple:

Field1	Field2	Field3	…

Fields and Tuples are Fixed Width!

SeqScan (Implements DbIterator)

- DbIterator class implemented by all operators
  - open()
  - close()
  - getTupleDesc()
  - hasNext()
  - next()
  - rewind()
- Iterator model: chain iterators together
  - Use DbFileIterator from HeapFile

```
// construct a 3-column table schema
Type types[] = new Type[]{ Type.INT_TYPE, Type.INT_TYPE, Type.INT_TYPE };
String names[] = new String[]{ "field0", "field1", "field2" };
TupleDesc descriptor = new TupleDesc(types, names);

// create the table, associate it with some_data_file.dat
// and tell the catalog about the schema of this table.
HeapFile table1 = new HeapFile(new File("some_data_file.dat"), descriptor);
Database.getCatalog().addTable(table1);

// construct the query: we use a simple SeqScan, which spoonfeeds
// tuples via its iterator.
TransactionId tid = new TransactionId();
SeqScan f = new SeqScan(tid, table1.id());

// and run it
f.open();
while (f.hasNext()) {
        Tuple tup = f.next();
        System.out.println(tup);
}
f.close();
Database.getBufferPool().transactionComplete();
```

HeapFileEncoder.java

- Because you haven’t implemented insertTuple, you have no way to create data files
- HeapFileEncoder converts CSV files to HeapFiles
- Usage:
  - java -jar dist/simpledb.jar convert csv-file.txt numFields
- Produces a file csv-file.dat, that can be passed to HeapFile constructor.

Compiling, Testing, and Running

- Demo on running tests and debugging in Ubuntu and with Eclipse