**CSI 508 - Database Systems I - Spring 2017**

**Programming Assignment I**

**Rujit Raval**

**Student ID#: 001319222**

**Part 1. Fields and Tuples**

**PAGE: TupleDisc.java**

public Type getType(int i) throws NoSuchElementException {

// some code goes here

if (i>=0 && i<types.length)

return types[i];

throw new NoSuchElementException("Not a valid value");

}

public String toString() {

// some code goes here

String OutputString="";

int i;

for(i=0;i<types.length-1;i++)

{

OutputString=OutputString+types[i]+"("+names[i]+"), ";

}

OutputString=OutputString+types[i]+"("+names[i]+")";

return OutputString;

//throw new UnsupportedOperationException("Implement this");

}

**PAGE: Tuple.java**

public String toString() {

// some code goes here

String OutputString ="";

int i;

for(i = 0;i<fields.length-1;i++){

OutputString = OutputString + fields[i]+",";

}

OutputString = OutputString + fields[i];

return OutputString;

//throw new UnsupportedOperationException("Implement this");

}

**Part 2. Catalog**

**PAGE: Catalog.java**

public int getTableId(String name) {

// some code goes here

if(name2tableID.get(name)!= null)

return(name2tableID.get(name));

throw new NoSuchElementException("There is no elements in the table.");

}

public TupleDesc getTupleDesc(int tableid) throws NoSuchElementException {

// some code goes here

if(tableID2desc.get(tableid)!= null)

return tableID2desc.get(tableid);

throw new NoSuchElementException("The table does not exists !");

}

public DbFile getDbFile(int tableid) throws NoSuchElementException {

// some code goes here

if(tableID2dbFile.get(tableid)!= null)

return tableID2dbFile.get(tableid);

throw new NoSuchElementException("The table does not exists !");

}

**Part 3. BufferPool**

**PAGE: BufferPool.java**

public Page getPage(TransactionId tid, PageId pid, Permissions perm)

throws TransactionAbortedException, DbException {

Page page = pages.get(pid);

if (pages.size() > numPages)

throw new UnsupportedOperationException("Maximum limit reached !");

else

{

if(pages.contains(pid))

return page;

else

{

DbFile db = Database.getCatalog().getDbFile(pid.getTableId());

db.readPage(pid);

}

throw new UnsupportedOperationException("Can not handle this operation !");

}

// some code goes here

//throw new UnsupportedOperationException("Implement this");

}

**Part 4. HeapPage access**

**PAGE: HeapPage.java**

public Tuple getTuple(int entryID) {

// some code goes heres

if(entryID < entryCount())

{

DataInputStream in=new DataInputStream(new ByteArrayInputStream(data,tupleLocation(entryID),data.length));

Tuple t=createTuple(in);

t.setRecordId(new RecordId(pid, entryID));

return t;

}

else

return null;

//throw new UnsupportedOperationException("Implement this");

}

protected int entryCount() {

// some code goes here

return readInt(data, 0);

//throw new UnsupportedOperationException("Implement this");

}

protected int tupleLocation(int entryID) {

// some code goes here

return readInt(data,4 + 4 \* entryID);

//throw new UnsupportedOperationException("Implement this");

}

public Iterator<Tuple> iterator() {

int i = 0;

while( i != entryCount())

{

final Iterator<Tuple> new\_iterator = new Iterator<Tuple>() {

@Override

public Tuple next() {

int entryID=0;

// TODO Auto-generated method stub

while (hasNext())

{

entryID++;

Tuple T = getTuple(entryID);

if(T == null)

throw new UnsupportedOperationException("Tuple is deleted !");

else

return T;

}

return null;

}

@Override

public boolean hasNext() {

if(entryCount() != 0){

return true;

}

else{

return false;

}

}

};

}

return null;

//throw new UnsupportedOperationException("Implement this");

}

**Part 5. HeapFile access**

**PAGE: HeapFile.java**

public Page readPage(PageId pid) {

// some code goes here

byte[] buffer = new byte[BufferPool.PAGE\_SIZE];

FileInputStream finput = null;

HeapPage hpage = null;

try {

finput = new FileInputStream(file);

finput.skip(pid.pageno() \* BufferPool.PAGE\_SIZE);

finput.read(buffer);

finput.close();

hpage = new HeapPage((HeapPageId)pid, buffer);

}

catch (IOException e)

{

throw new IllegalArgumentException("Page is not available !");

}

return hpage;

//throw new UnsupportedOperationException("Implement this");

}

public boolean hasNext() throws DbException, TransactionAbortedException {

// some code goes here

if (iterator == null) {

return false;

} else if (iterator.hasNext()) {

return true;

} else

{

nextPageID++;

}

throw new UnsupportedOperationException("Implement this");

}

@Override

public Tuple next() throws DbException, TransactionAbortedException, NoSuchElementException {

// some code goes here

if (hasNext())

return iterator.next();

else

throw new NoSuchElementException();

//throw new UnsupportedOperationException("Implement this");

}

**Part 6. HeapPage Mutability**

**PAGE: HeapPage.java**

public void deleteTuple(Tuple t) throws DbException {

// some code goes here

int x = t.getRecordId().tupleno();

saveTupleLocation(-1, x);

throw new UnsupportedOperationException("Implement this");

}

public void addTuple(Tuple t) throws DbException {

// some code goes here

byte[] b = toByteArray(t);

if(!(freeSpaceSize() < b.length + 4))

{

int location = endOfFreeSpace() - b.length;

System.arraycopy(b, 0, data, location, b.length);

saveEntryCount(location);

saveTupleLocation(b.length, location);

t.setRecordId(new RecordId(pid, location));

}

else

{

throw new DbException("Page can not be added");

}

throw new UnsupportedOperationException("Implement this");

}

Total Time spend is about 2-3 days.